Guía del usuario

AWS Glue

AWS Glue: Guía del usuario

Table of Contents

¿Qué es AWS Glue? 1
Características de AWS Glue 2
Conozca las innovaciones en AWS Glue 4
Introducción a AWS Glue 4
Acceso a AWS Glue 4
Servicios relacionados 5
Cómo funciona 6
Los trabajos de ETL sin servidor se ejecutan de forma aislada 7
Conceptos 8
Terminología de AWS Glue 10
Componentes 13
Consola de AWS Glue 13
AWS Glue Data Catalog 14
Rastreadores y clasificadores de AWS Glue 15
Operaciones de ETL de AWS Glue 15
ETL de streaming en AWS Glue 16
El sistema de trabajos de AWS Glue 16
Componentes de ETL visuales 16
AWS Glue para Spark y AWS Glue para Ray 22
¿Qué es AWS Glue para Ray? 23
Conversión de esquemas semiestructurados a esquemas relacionales 24
AWS Tipos de pegamentos 26
AWS Tipos de catálogos de datos de Glue 26
Escribe guiones en AWS Glue with Spark 26
AWS Tipos de Glue Crawler 27
Introducción 28
Información general para el uso de AWS Glue 28
Configuración de permisos de IAM 30
Siguientes pasos 35
Permisos de IAM para usar operaciones de ETL visuales 35
Introducción a los cuaderno en AWS Glue Studio 47
Configuración de perfiles de uso 49
Administrar los perfiles de uso 51
Perfiles de uso y trabajos 63

Introducción al AWS Glue Data Catalog 64
Información general 64
Paso 1: Crear una base de datos 64
Paso 2. Creación de una tabla 66
Siguientes pasos 67
Configuración del acceso de red a los almacenes de datos 71
Configuración de una VPC para conectarse a PyPI para AWS Glue 72
Configuración de DNS en la VPC 74
Configuración del cifrado 75
Configuración de redes para desarrollo 79
Configuración de su red para un punto de conexión de desarrollo 79
Configuración de Amazon EC2 para el servidor de blocs de notas 81
Detección y catalogación de datos 83
Cómo completar el Catálogo de datos 86
Uso de Rastreador de AWS Glue 86
Cómo definir los metadatos manualmente 192
Integración con otros servicios de AWS 212
Configuración del Catálogo de datos 213
Cómo completar y administrar las tablas transaccionales 216
Creación de tablas de Iceberg 216
Optimización de las tablas de Iceberg 220
Administración del Catálogo de datos 233
Cómo actualizar el esquema y añadir nuevas particiones 234
Cómo optimizar el rendimiento de las consultas con las estadísticas de columnas 241
Cifrado del Catálogo de datos 254
Cómo proteger su Catálogo de datos con Lake Formation 254
Cómo acceder al Catálogo de datos 254
Prácticas recomendadas para el uso del Catálogo de datos 255
AWS Glue Schema Registry 257
Schemas 258
Registries 260
Compatibilidad y control de versiones de esquemas 261
Bibliotecas Serde de código abierto 267
Cuotas del Schema Registry 267
Cómo funcionan 268
Introducción 270

Integración con AWS Glue Schema Registry ... 293
Migración a AWS Glue Schema Registry ... 319
Conexión a datos ... 322
Propiedades de las conexiones de AWS Glue ... 323
Propiedades de conexión requeridas ... 324
Propiedades de las conexiones JDBC ... 325
Propiedades de conexión de MongoDB y MongoDB Atlas ... 330
Propiedades de conexión de Salesforce ... 331
Conexión Snowflake ... 332
Conexión vertical ... 333
Conexión a SAP HANA ... 334
Conexión a Azure SQL ... 335
Conexión de Teradata Vantage ... 335
OpenSearch Conexión de servicio ... 336
Conexión de Azure Cosmos ... 337
Propiedades de las conexiones SSL ... 338
Propiedades de las conexiones de Kafka para la autenticación de clientes ... 341
BigQuery Conexión a Google ... 342
Conexión vertical ... 333
Almacenamiento de credenciales de conexión en AWS Secrets Manager ... 342
Adición de una conexión de AWS Glue ... 343
Conexión a Redshift ... 344
Conexión a Azure Cosmos DB ... 348
Conexión a Azure SQL ... 352
Conectándose a BigQuery ... 355
Conexión a MongoDB ... 360
Conexión al servicio OpenSearch ... 364
Conectarse a Salesforce ... 367
Conexión a SAP HANA ... 379
Conectarse a Snowflake ... 383
Conexión a Teradata ... 387
Conexión a Vertica ... 391
Uso de conectores y conexiones ... 395
Conexión con orígenes de datos ... 426
Agregar una conexión JDBC con sus propios controladores JDBC ... 435
Prueba de una conexión de AWS Glue ... 439

Configuración de las llamadas de AWS para que pasen a través de la VPC 440
Conexión a un almacén de datos de JDBC en una VPC 441
Acceso a datos de VPC mediante interfaces de red elásticas 442
Propiedades de la interfaz de red elástica 443
Uso de una conexión MongoDB o MongoDB Atlas 443
Rastreo de un almacén de datos de Amazon S3 mediante un punto de conexión de VPC 444
Requisitos previos 444
Crear la conexión a Amazon S3 446
Prueba de la conexión a Amazon S3 448
Creación de un rastreador para un almacén de datos de Amazon S3 450
Creación de un rastreador para tablas del Catálogo de datos respaldadas por Amazon S3 . 452
Ejecución de un rastreador 453
Resolución de problemas 453
Solución de problemas de conexión 453
Tutorial: uso de AWS Glue Connector for Elasticsearch 454
Requisitos previos 455
Paso 1: (opcional) cree un secreto de AWS para la información del clúster de OpenSearch 455
Paso 2: suscríbase al conector 456
Paso 3: activar el conector en AWS Glue Studio y crear una conexión 457
Paso 4: configurar un rol de IAM para el trabajo de ETL 458
Paso 5: crear un trabajo que utilice la conexión OpenSearch 459
Paso 6: ejecutar el trabajo 460
Creación de trabajos de AWS Glue con sesiones interactivas 461
Información general sobre las sesiones interactivas de AWS Glue 461
Limitaciones 462
Introducción a las sesiones interactivas de AWS Glue 462
Requisitos previos para configurar las sesiones interactivas de manera local 462
Instalación de Jupyter y sesiones AWS Glue interactivas (kernels de Jupyter) 462
Ejecución de Jupyter 463
Configuración de credenciales de sesión y región 463
Actualización desde la versión preliminar de las sesiones interactivas 465
Uso de las sesiones interactivas con SageMaker Studio 465
Uso de las sesiones interactivas con Microsoft Visual Studio Code 465
Configuración de las sesiones interactivas de AWS Glue para cuadernos de Jupyter y AWS Glue Studio 469

Introducción a los comandos mágicos de Jupyter 469
Comandos mágicos compatibles con las sesiones interactivas de AWS Glue para Jupyter .. 469
Denominación de sesiones 488
Especificación de un rol de IAM para las sesiones interactivas 489
Configuración de sesiones con perfiles con nombre 490
AWS Glue para las sesiones interactivas de Ray (versión preliminar) 491
Sesiones interactivas de Ray en la AWS Glue Studio consola 491
Sesiones interactivas de Ray mediante el uso del kernel de Jupyter 492
Valores predeterminados del tiempo de espera de la sesión interactiva de Ray 492
Comandos mágicos compatibles con las sesiones interactivas de AWS Glue Ray 493
Sesiones interactivas con IAM 494
Entidades principales de IAM que se utilizan con sesiones interactivas 495
Configuración de una entidad principal del cliente 495
Configuración de un rol de tiempo de ejecución 496
Haga que su sesión sea privada con TagOnCreate 497
Consideraciones respecto de la política de IAM 502
Conversión de un script o cuaderno en un trabajo de AWS Glue 503
Sesiones interactivas de AWS Glue para streaming 503
Cambio del tipo de sesión de streaming 503
Muestreo de flujo de entrada para desarrollo interactivo 503
Ejecución de aplicaciones de streaming en las sesiones interactivas 505
Desarrollo y pruebas a nivel local 506
Desarrollo mediante AWS Glue Studio 507
Desarrollo mediante sesiones interactivas 507
Desarrollo mediante una imagen de Docker 507
Desarrollo mediante la biblioteca de ETL de AWS Glue 519
Puntos de conexión de desarrollo 527
Migración de puntos de conexión de desarrollo a sesiones interactivas 529
Desarrollo de scripts usando puntos de conexión de desarrollo 531
Administración de blocs de notas 560
Creación de trabajos de ETL visuales con AWS Glue Studio 562
Inicie sesión en la consola 562
Siguientes pasos para crear un trabajo en AWS Glue Studio 563
Operaciones de ETL visuales con AWS Glue Studio 563
Empezar trabajos en AWS Glue Studio 563
Características del editor de trabajo 565

Edición de nodos de transformación de datos administrados por AWS Glue 573
Transformaciones visuales personalizadas 640
Uso de marcos de lagos de datos con AWS Glue Studio 659
Configuración de nodos de destino de datos 671
Edición o carga de un script de trabajo 676
Cambio de los nodos principales de un nodo en el diagrama de trabajo 681
Eliminación de nodos del diagrama de trabajo 682
Añadir parámetros de origen y destino al nodo AWS Glue del Catálogo de datos 686
Uso de los sistemas de control de versiones de Git en AWS Glue 688
Creación de código con cuadernos de AWS Glue Studio 697
Información general sobre el uso de cuaderno 698
Creación de un trabajo de ETL mediante cuaderno en AWS Glue Studio 699
Componentes del editor de cuaderno 700
Cómo guardar el cuaderno y el script de trabajo 701
Administración de las sesiones de cuaderno 702
Usar CodeWhisperer con AWS Glue Studio notebooks 704
Vista de las ejecuciones de trabajo 704
Acceso al panel de monitoreo de trabajos 704
Información general del panel de monitoreo de trabajos 704
Vista de las ejecuciones de trabajo 705
Visualización de los registros de ejecución de trabajo 710
Visualización de los detalles de una ejecución de trabajo 711
Visualización de métricas de Amazon CloudWatch para una ejecución de trabajo de Spark 714
Visualización de métricas de Amazon CloudWatch para una ejecución de trabajo de Ray ... 715
Detectar y procesar información confidencial 717
Elegir cómo desea que se escaneen los datos 717
Elección de las entidades de PII que se desea detectar 719
Especificar el nivel de sensibilidad de detección 723
Elegir qué hacer con los datos de PII identificados 724
Agregar anulaciones de acciones detalladas 725
Administración de trabajos 726
Iniciar una ejecución de trabajo 726
Programar ejecuciones de trabajo 727
Administrar programaciones de trabajo 728
Detener ejecuciones de trabajo 729

Ver los trabajos ... 729
Ver información sobre las ejecuciones de trabajos recientes ... 730
Ver el script de trabajo ... 731
Modificar las propiedades del trabajo ... 732
Guardar el trabajo ... 735
Clonación de un trabajo ... 737
Eliminación de trabajos ... 737
Trabajar con tareas ... 739
Versiones de AWS Glue ... 739
Versiones de AWS Glue ... 739
Ejecución de trabajos de ETL de Spark con tiempos de inicio reducidos ... 757
Migración de trabajos de AWS Glue para Spark a la versión 3.0 de AWS Glue ... 762
Migración de trabajos de AWS Glue para Spark a la versión 4.0 de AWS Glue ... 771
Migración de AWS Glue para Ray (vista previa) a AWS Glue para Ray ... 787
Política de compatibilidad de versiones de AWS Glue ... 788
Trabajar con trabajos de Spark ... 790
Parámetros del flujo de trabajo ... 790
Spark y PySpark empleos ... 800
Trabajos ETL de streaming ... 947
Coincidencia de registros con FindMatches ... 962
Migrar programas Spark ... 999
Trabajar con tareas de Ray ... 1008
Introducción a AWS Glue para Ray ... 1008
Entornos de tiempo de ejecución de Ray compatibles ... 1009
Contabilidad de los trabajadores en los trabajos de Ray ... 1010
Parámetros de los trabajos de Ray ... 1011
Métricas de trabajo de Ray ... 1014
Configuración de las propiedades de trabajos del intérprete de comandos de Python ... 1016
Limitaciones ... 1016
Definición de las propiedades de trabajos de shell de Python ... 1016
Bibliotecas compatibles con trabajos de shell de Python ... 1018
Proporcionar su propia biblioteca de Python ... 1020
Use AWS CloudFormation con los trabajos del intérprete de comandos de Python en AWS Glue ... 1024
Supervisión ... 1024
AWS etiquetas ... 1026

Automatización con CloudWatch Events ... 1031
Monitoreo de recursos AWS Glue ... 1033
Registro con CloudTrail ... 1036
Estados de ejecuciones de trabajos ... 1039
AWS Glue Streaming ... 1043
Casos de uso de la transmisión ... 1043
¿Cuáles son las ventajas de usar AWS Glue Streaming? ... 1044
¿Cuándo usar AWS Glue Streaming? ... 1045
Orígenes de datos admitidos ... 1046
Destinos de datos admitidos ... 1046
Tutorial: Cree su primera carga de trabajo de transmisión con AWS Glue Studio ... 1047
Requisitos previos ... 1047
Consumir datos de transmisión desde Amazon Kinesis ... 1047
Tutorial: Cree su primera carga de trabajo de transmisión con los cuadernos de AWS Glue Studio ... 1058
Requisitos previos ... 1059
Consumir datos de transmisión desde Amazon Kinesis ... 1059
Conceptos de la transmisión ... 1066
Anatomía de un trabajo de transmisión de AWS Glue ... 1067
Conexiones de Kafka ... 1070
Conexión de Kinesis ... 1077
Opciones de transmisión ... 1084
Escalado automático de la transmisión de AWS Glue ... 1085
Habilitar Auto Scaling en AWS Glue Studio ... 1085
Habilitación de Auto Scaling con AWS CLI o SDK ... 1086
Cómo funcionan ... 1087
Periodos de mantenimiento ... 1089
Configuración de una ventana de mantenimiento ... 1089
El comportamiento de la ventana de mantenimiento ... 1090
Supervisión de trabajos ... 1091
Gestión de la pérdida de datos ... 1093
Conceptos avanzados de la transmisión de AWS Glue ... 1094
Consideraciones de tiempo a la hora de procesar los trabajos de transmisión ... 1094
Creación de ventanas ... 1095
Manejo de datos tardíos y marcas de agua ... 1101
Supervisión de los trabajos de transmisión de AWS Glue ... 1103

Visualización de métricas 1104
Análisis profundo de las métricas 1105
Cómo obtener el mejor rendimiento 1110
AWS Glue Calidad de los datos 1112
Beneficios y características principales 1112
Funcionamiento 1113
Calidad de los datos para AWS Glue Data Catalog 1113
Calidad de datos para trabajos de AWS Glue ETL 1114
Comparación de los puntos de partida de calidad de los AWS Glue datos 1114
Consideraciones 1116
Terminología 1116
Límites 1117
Notas de publicación sobre la calidad de los datos AWS Glue 1118
Disponibilidad general: características nuevas 1118
27 de noviembre de 2023 (Vista previa) 1118
12 de marzo de 2024 1119
26 de junio de 2024 1119
Detección de anomalías en Calidad de los datos AWS Glue 1119
Funcionamiento 1120
Uso de analizadores para inspeccionar los datos 1120
Uso de la DetectAnomaly regla 1121
Ventajas y casos de uso de la detección de anomalías 1121
Permisos de IAM para Calidad de los datos de AWS Glue 1123
Permisos de IAM 1123
Se requiere una configuración de IAM para programar las ejecuciones de evaluación 1125
Ejemplos de políticas de IAM 1127
Introducción a Calidad de datos de AWS Glue para el Data Catalog 1130
Requisitos previos 1131
Ejemplo paso a paso 1131
Generar recomendaciones de reglas 1131
Recomendaciones de reglas de monitoreo 1133
Edición de los conjuntos de reglas recomendados 1134
Creación de un nuevo conjunto de reglas 1135
Ejecute un conjunto de reglas para evaluar la calidad de los datos 1137
Visualice la puntuación de la calidad de los datos y los resultados 1138
Temas relacionados de 1139

Evaluación de la calidad de los datos con AWS Glue Studio 1139
Ventajas 1140
Evaluación de la calidad de los datos para los trabajos de ETL en AWS Glue Studio 1140
Generador de reglas de Calidad de datos 1146
Configuración de la detección de anomalías y generación de información 1151
Calidad de datos para trabajos de ETL en las libretas AWS Glue Studio 1156
Requisitos previos 1157
Crear un trabajo de ETL en AWS Glue Studio 1157
Referencia del lenguaje de definición de calidad de datos (DQDL) 1162
Sintaxis 1164
Referencia de tipo de regla 1178
Uso de las API para medir y gestionar la calidad de los datos 1224
Requisitos previos 1225
Cómo trabajar con las recomendaciones de Calidad de datos de AWS Glue 1225
Trabajar con los conjuntos de reglas de Calidad de datos de AWS Glue 1228
Cómo trabajar con ejecuciones de Calidad de datos de AWS Glue 1230
Cómo trabajar con los resultados de Calidad de datos de AWS Glue 1235
Configuración de alertas, implementaciones y programación 1236
Configuración de alertas y notificaciones en la EventBridge integración de Amazon 1236
Configure alertas y notificaciones en la CloudWatch integración 1244
Cómo consultar de resultados de calidad de datos 1246
Implementación de reglas de calidad de datos 1250
Programación de reglas de calidad de datos 1250
Solución de errores de calidad de datos de AWS Glue 1250
Error: falta el módulo 1251
Error: permisos insuficientes 1251
Error: los conjuntos de reglas no son únicos 1252
Error: tablas con caracteres especiales 1252
Error: desbordamiento con un conjunto de reglas grande 1252
Error: falló el estado de la regla 1252
AnalysisException: No se ha podido comprobar la existencia de la base de datos predeterminada 1253
El mapa de teclas dado no es adecuado para determinados marcos de datos 1253
java.lang. RuntimeException : No se pudieron recuperar los datos. 1254
ERROR DE INICIO: error al descargar desde S3 para bucket 1254
InvalidInputException (estado: 400): DataQuality las reglas no se pueden analizar 1254

Error: EventBridge no activa los trabajos de calidad de datos de Glue según la programación que configuré ... 1255
Errores de CustomSQL ... 1255
Reglas dinámicas ... 1256
Excepción en la clase de usuario: org.apache.spark.sql. AnalysisException: org.apache.hadoop.hive.ql.metadata. HiveException ... 1258
UNCLASSIFIED_ERROR; IllegalArgumentException: Error de análisis: no se proporcionaron reglas ni analizadores., no hay una alternativa viable en la entrada ... 1259
Integración de datos de Amazon Q en AWS Glue ... 1260
¿Qué es Amazon Q? ... 1260
Integración de datos de Amazon Q en AWS Glue ... 1260
Trabajo con la integración de datos de Amazon Q ... 1261
Prácticas recomendadas ... 1263
Mejora del servicio ... 1264
Consideraciones ... 1264
Cómo configurar la integración de datos de Amazon Q ... 1264
Configuración de permisos de IAM ... 1264
Generación de código compatible ... 1267
Interacciones de muestra ... 1268
Interacciones de chat de Amazon Q ... 1268
AWS Glue Interacciones del cuaderno de Studio ... 1270
Orquestación ... 1274
Inicio de trabajos y rastreadores mediante desencadenadores ... 1274
Disparadores de AWS Glue ... 1274
Agregado de desencadenadores ... 1277
Activación y desactivación de desencadenadores ... 1281
Realización de actividades de ETL complejas mediante esquemas y flujos de trabajo ... 1282
Descripción general de flujos de trabajo ... 1283
Creación y desarrollo manual de un flujo de trabajo ... 1287
Inicio de un flujo de trabajo con un evento de EventBridge ... 1292
Visualizar los eventos de EventBridge que iniciaron un flujo de trabajo ... 1299
Ejecución y monitoreo de un flujo de trabajo ... 1301
Detener una ejecución de flujo de trabajo ... 1303
Reparar y reanudar la ejecución de un flujo de trabajo ... 1304
Obtención y configuración de propiedades de ejecución de flujo de trabajo ... 1311
Consulta de flujos de trabajo mediante la AWS Glue API ... 1312

Restricciones de esquemas y flujos de trabajo 1317
Solución de errores de esquema 1318
Permisos de personas y roles para esquemas 1323
Desarrollo de esquemas 1327
Información general de los esquemas 1328
Desarrollo de esquemas 1332
Registrar un esquema 1357
Visualización de esquemas 1359
Actualizar un proyecto 1361
Creación de un flujo de trabajo a partir de un esquema 1363
Visualización de ejecuciones de esquemas 1366
AWS CloudFormation para AWS Glue 1367
Base de datos de muestra 1369
Base de datos, tabla y particiones de muestra 1370
Clasificador de Grok de muestra 1374
Clasificador de JSON de muestra 1375
Clasificador de XML de muestra 1376
Ejemplo de rastreador de Amazon S3 1377
Conexión de muestra 1380
Rastreador de JDBC de muestra 1381
Trabajo para Amazon S3 de ejemplo a Amazon S3 1384
Trabajo de ejemplo para JDBC a Amazon S3 1385
Disparador bajo demanda de muestra 1387
Disparador programado de muestra 1388
Disparador condicional de muestra 1389
Ejemplo de transformación de machine learning 1391
Ejemplo de conjunto de reglas de calidad de los datos 1392
Ejemplo de conjunto de reglas de calidad de datos con el programador de EventBridge 1393
Punto de conexión de desarrollo de muestra 1396
Guía de programación de AWS Glue 1398
Suministro de sus propios scripts personalizados 1398
Glue para Spark AWS 1399
Tutorial: Escritura de un script de Spark 1400
ETL en PySpark 1413
ETL en Scala 1647
Características y optimizaciones 1734

AWS Glue para Ray 1995
Tutorial: Cómo escribir un script de Ray 1995
Uso de Ray Core y Ray Data en AWS Glue para Ray 2001
Proporcionar archivos y bibliotecas de Python 2003
Conexión a datos 2009
Trabajando con los AWS SDK 2011
AWS Glue API 2013
Seguridad 2035
— tipos de datos — 2035
DataCatalogEncryptionSettings 2036
EncryptionAtRest 2036
ConnectionPasswordEncryption 2037
EncryptionConfiguration 2038
S3Encryption 2038
CloudWatchEncryption 2038
JobBookmarksEncryption 2039
SecurityConfiguration 2039
Política de Glue 2040
— operaciones — 2040
GetDataCatalogEncryptionSettings (get_data_catalog_encryption_settings) 2041
PutDataCatalogEncryptionSettings (put_data_catalog_encryption_settings) 2041
PutResourcePolicy (put_resource_policy) 2042
GetResourcePolicy (get_resource_policy) 2043
DeleteResourcePolicy (delete_resource_policy) 2044
CreateSecurityConfiguration (create_security_configuration) 2045
DeleteSecurityConfiguration (delete_security_configuration) 2046
GetSecurityConfiguration (get_security_configuration) 2047
GetSecurityConfigurations (get_security_configurations) 2047
GetResourcePolicies (get_resource_policies) 2048
Catálogo 2049
Bases de datos 2050
Tablas 2059
Particiones 2100
Conexiones 2126
Funciones definidas por el usuario 2145
Importación de un catálogo de Athena 2152

Optimizador de tablas 2154
— data types — 2154
TableOptimizer 2154
TableOptimizerConfiguration 2154
TableOptimizerRun 2155
RunMetrics 2155
BatchGetTableOptimizerEntry 2156
BatchTableOptimizer 2157
BatchGetTableOptimizerError 2157
— operaciones — 2158
GetTableOptimizer (get_table_optimizer) 2158
BatchGetTableOptimizer (batch_get_table_optimizer) 2159
ListTableOptimizerRuns (list_table_optimizer_runs) 2160
CreateTableOptimizer (create_table_optimizer) 2161
DeleteTableOptimizer (delete_table_optimizer) 2163
UpdateTableOptimizer (update_table_optimizer) 2163
Rastreadores y clasificadores 2165
Clasificadores 2165
Rastreadores 2179
Estadísticas de las columnas 2207
Programador 2215
Generación automática de scripts de ETL 2218
— tipos de datos — 2218
CodeGenNode 2218
CodeGenNodeArg 2219
CodeGenEdge 2219
Ubicación 2220
CatalogEntry 2220
MappingEntry 2220
— operaciones — 2221
CreateScript (create_script) 2221
GetDataflowGraph (get_dataflow_graph) 2222
GetMapping (get_mapping) 2223
GetPlan (get_plan) 2223
API de Visual Job 2225
— data types — 2225

CodeGenConfigurationNode 2229
JDBC ConnectorOptions 2235
StreamingDataPreviewOptions 2237
AthenaConnectorSource 2237
JDBC ConnectorSource 2238
SparkConnectorSource 2239
CatalogSource 2239
MySQL CatalogSource 2240
PostgresQL CatalogSource 2240
OracleSQL CatalogSource 2241
Microsoft SQL ServerCatalogSource 2241
CatalogKinesisSource 2241
DirectKinesisSource 2242
KinesisStreamingSourceOptions 2243
CatalogKafkaSource 2246
DirectKafkaSource 2246
KafkaStreamingSourceOptions 2247
RedshiftSource 2250
AmazonRedshiftSource 2250
AmazonRedshiftNodeData 2250
AmazonRedshiftAdvancedOption 2253
Opción 2253
S3 CatalogSource 2254
S3 SourceAdditionalOptions 2254
S3 CsvSource 2255
DirectJDBCSource 2257
S3 DirectSourceAdditionalOptions 2258
S3 JsonSource 2258
S3 ParquetSource 2260
S3 DeltaSource 2262
S3 CatalogDeltaSource 2262
CatalogDeltaSource 2263
S3 HudiSource 2264
S3 CatalogHudiSource 2264
CatalogHudiSource 2265
DynamoDB CatalogSource 2266

RelationalCatalogSource 2266
JDBC ConnectorTarget 2266
SparkConnectorTarget 2267
BasicCatalogTarget 2268
MySQL CatalogTarget 2269
PostgresQL CatalogTarget 2269
OracleSQL CatalogTarget 2270
Microsoft SQL ServerCatalogTarget 2270
RedshiftTarget 2271
AmazonRedshiftTarget 2272
UpsertRedshiftTargetOptions 2272
S3 CatalogTarget 2272
S3 GlueParquetTarget 2273
CatalogSchemaChangePolicy 2274
S3 DirectTarget 2274
S3 HudiCatalogTarget 2275
S3 HudiDirectTarget 2276
S3 DeltaCatalogTarget 2277
S3 DeltaDirectTarget 2278
DirectSchemaChangePolicy 2279
ApplyMapping 2279
Correspondencia 2280
SelectFields 2281
DropFields 2281
RenameField 2282
Spigot 2282
Join 2283
JoinColumn 2284
SplitFields 2284
SelectFromCollection 2284
FillMissingValues 2285
Filtro 2285
FilterExpression 2286
FilterValue 2286
CustomCode 2287
SparkSQL 2287

SqlAlias 2288
DropNullFields 2289
NullCheckBoxList 2289
NullValueField 2290
Tipo de datos 2290
Merge 2291
Unión 2291
PIIDetection 2292
Agregado 2293
DropDuplicates 2293
GovernedCatalogTarget 2294
GovernedCatalogSource 2295
AggregateOperation 2295
GlueSchema 2296
GlueStudioSchemaColumn 2296
GlueStudioColumn 2296
DynamicTransform 2297
TransformConfigParameter 2298
EvaluateDataQuality 2299
DQ ResultsPublishingOptions 2300
DQ StopJobOnFailureOptions 2300
EvaluateDataQualityMultiFrame 2300
Receta 2302
RecipeReference 2302
SnowflakeNodeData 2302
SnowflakeSource 2305
SnowflakeTarget 2305
ConnectorDataSource 2306
ConnectorDataTarget 2307
Jobs 2307
Trabajos 2308
Ejecuciones de trabajo 2335
Desencadenadores 2354
Sesiones interactivas 2368
— data types — 2368
Sesión 2368

SessionCommand 2371
Instrucción 2371
StatementOutput 2372
StatementOutputData 2373
ConnectionsList 2373
— operaciones — 2373
CreateSession (create_sesión) 2374
StopSession (stop_session) 2377
DeleteSession (delete_session) 2378
GetSession (get_session) 2379
ListSessions (list_sesiones) 2380
RunStatement (declaración_ejecución) 2381
CancelStatement (declaración_cancelación) 2382
GetStatement (get_statement) 2383
ListStatements (list_declaraciones) 2384
DevEndpoints 2385
— tipos de datos — 2385
DevEndpoint 2385
DevEndpointCustomLibraries 2389
— operaciones — 2390
CreateDevEndpoint (create_dev_endpoint) 2390
UpdateDevEndpoint (update_dev_endpoint) 2396
DeleteDevEndpoint (delete_dev_endpoint) 2398
GetDevEndpoint (get_dev_endpoint) 2398
GetDevEndpoints (get_dev_endpoints) 2399
BatchGetDevEndpoints (batch_get_dev_endpoints) 2400
ListDevEndpoints (list_dev_endpoints) 2401
Schema Registry 2402
— data types — 2402
RegistryId 2403
RegistryListItem 2403
MetadataInfo 2404
OtherMetadataValueListItem 2404
SchemaListItem 2405
SchemaVersionListItem 2405
MetadataKeyValuePair 2406

SchemaVersionErrorItem 2406
ErrorDetails 2407
SchemaVersionNumber 2407
SchemaId 2407
— operaciones — 2408
CreateRegistry (create_registry) 2409
CreateSchema (create_schema) 2410
GetSchema (get_schema) 2414
ListSchemaVersions (list_schema_versions) 2416
GetSchemaVersion (get_schema_version) 2418
GetSchemaVersionsDiff (get_schema_versions_diff) 2419
ListRegistries (list_registros) 2420
ListSchemas (list_esquemas) 2421
RegisterSchemaVersion (register_schema_version) 2422
UpdateSchema (update_schema) 2424
CheckSchemaVersionValidity (check_schema_version_valid) 2426
UpdateRegistry (update_registry) 2427
GetSchemaByDefinition (get_schema_by_definition) 2428
GetRegistry (get_registry) 2429
PutSchemaVersionMetadata (put_schema_version_metadata) 2430
QuerySchemaVersionMetadata (query_schema_version_metadata) 2432
RemoveSchemaVersionMetadata (remove_schema_version_metadata) 2433
DeleteRegistry (delete_registry) 2435
DeleteSchema (delete_schema) 2436
DeleteSchemaVersions (delete_schema_versions) 2437
Flujos de trabajo 2438
— tipos de datos — 2438
JobNodeDetails 2439
CrawlerNodeDetails 2439
TriggerNodeDetails 2439
Rastreo 2439
Nodo 2440
Periferia 2441
Flujo de trabajo 2441
WorkflowGraph 2443
WorkflowRun 2443

WorkflowRunStatistics 2445
StartingEventBatchCondition 2446
Proyecto 2446
BlueprintDetails 2447
LastActiveDefinition 2448
BlueprintRun 2448
— operaciones — 2450
CreateWorkflow (create_workflow) 2450
UpdateWorkflow (update_workflow) 2452
DeleteWorkflow (delete_workflow) 2453
GetWorkflow (get_workflow) 2454
ListWorkflows (list_workflows) 2454
BatchGetWorkflows (batch_get_workflows) 2455
GetWorkflowRun (get_workflow_run) 2456
GetWorkflowRuns (get_workflow_runs) 2457
GetWorkflowRunProperties (get_workflow_run_properties) 2458
PutWorkflowRunProperties (put_workflow_run_properties) 2459
CreateBlueprint (create_blueprint) 2460
UpdateBlueprint (update_blueprint) 2461
DeleteBlueprint (delete_blueprint) 2462
ListBlueprints (list_blueprints) 2463
BatchGetBlueprints (batch_get_blueprints) 2464
StartBluePrintRun (start_blueprint_run) 2465
GetBlueprintRun (get_blueprint_run) 2466
GetBlueprintRuns (get_blueprint_runs) 2466
StartWorkflowRun (start_workflow_run) 2467
StopWorkflowRun (stop_workflow_run) 2468
ResumeWorkFlowRun (resume_workflow_run) 2469
Perfiles de uso 2470
— data types — 2470
ProfileConfiguration 2470
ConfigurationObject 2471
UsageProfileDefinition 2471
— operaciones — 2472
CreateUsageProfile (create_usage_profile) 2472
GetUsageProfile (get_usage_profile) 2473

UpdateUsageProfile (update_usage_profile) 2474
DeleteUsageProfile (delete_usage_profile) 2475
ListUsageProfiles (list_usage_profiles) 2476
Machine learning 2477
— tipos de datos — 2477
TransformParameters 2478
EvaluationMetrics 2478
MLTransform 2479
FindMatchesParameters 2482
FindMatchesMetrics 2483
ConfusionMatrix 2484
GlueTable 2485
TaskRun 2486
TransformFilterCriteria 2487
TransformSortCriteria 2488
TaskRunFilterCriteria 2489
TaskRunSortCriteria 2489
TaskRunProperties 2490
FindMatchesTaskRunProperties 2491
ImportLabelsTaskRunProperties 2491
ExportLabelsTaskRunProperties 2491
LabelingSetGenerationTaskRunProperties 2492
SchemaColumn 2492
TransformEncryption 2492
MLUserDataEncryption 2493
ColumnImportance 2493
— operaciones — 2494
CreateMLTransform (create_ml_transform) 2494
UpdateMLTransform (update_ml_transform) 2498
DeleteMLTransform (delete_ml_transform) 2501
GetMLTransform (get_ml_transform) 2501
GetMLTransforms (get_ml_transforms) 2505
ListMLTransforms (list_ml_transforms) 2506
StartMLEvaluationTaskRun (start_ml_evaluation_task_run) 2507
StartMLLabelingSetGenerationTaskRun (start_ml_labeling_set_generation_task_run) 2508
GetMLTaskRun (get_ml_task_run) 2509

GetMLTaskRuns (get_ml_task_runs) 2511
CancelMLTaskRun (cancel_ml_task_run) 2512
StartExportLabelsTaskRun (start_export_labels_task_run) 2513
StartImportLabelsTaskRun (start_import_labels_task_run) 2514
Calidad de datos 2516
— data types — 2516
DataSource 2517
DataQualityRulesetListDetails 2517
DataQualityTargetTable 2518
DataQualityRulesetEvaluationRunDescription 2518
DataQualityRulesetEvaluationRunFilter 2519
DataQualityEvaluationRunAdditionalRunOptions 2519
DataQualityRuleRecommendationRunDescription 2520
DataQualityRuleRecommendationRunFilter 2520
DataQualityResult 2521
DataQualityAnalyzerResult 2522
DataQualityObservation 2523
MetricBasedObservation 2523
DataQualityMetricValues 2524
DataQualityRuleResult 2524
DataQualityResultDescription 2525
DataQualityResultFilterCriteria 2526
DataQualityRulesetFilterCriteria 2526
— operaciones — 2527
StartDataQualityRulesetEvaluationRun (start_data_quality_ruleset_evaluation_run) 2528
CancelDataQualityRulesetEvaluationRun (cancel_data_quality_ruleset_evaluation_run) 2530
GetDataQualityRulesetEvaluationRun (get_data_quality_ruleset_evaluation_run) 2530
ListDataQualityRulesetEvaluationRuns (list_data_quality_ruleset_evaluation_runs) 2532
StartDataQualityRuleRecommendationRun (start_data_quality_rule_recommendation_run) 2533
CancelDataQualityRuleRecommendationRun
(cancel_data_quality_rule_recommendation_run) 2535
GetDataQualityRuleRecommendationRun (get_data_quality_rule_recommendation_run) .. 2535
ListDataQualityRuleRecommendationRuns (list_data_quality_rule_recommendation_runs) 2537
GetDataQualityResult (get_data_quality_result) 2538
BatchGetDataQualityResult (batch_get_data_quality_result) 2540
ListDataQualityResults (list_data_quality_results) 2541

CreateDataQualityRuleset (create_data_quality_ruleset) 2541
DeleteDataQualityRuleset (delete_data_quality_ruleset) 2543
GetDataQualityRuleset (get_data_quality_ruleset) 2543
ListDataQualityRulesets (conjuntos de reglas de calidad de lista de datos) 2545
UpdateDataQualityRuleset (update_data_quality_ruleset) 2546
Información confidencial 2547
— data types — 2547
CustomEntityType 2547
— operaciones — 2548
CreateCustomEntityType (create_custom_entity_type) 2548
DeleteCustomEntityType (delete_custom_entity_type) 2550
GetCustomEntityType (get_custom_entity_type) 2550
BatchGetCustomEntityTypes (batch_get_custom_entity_types) 2551
ListCustomEntityTypes (list_custom_entity_types) 2552
API de etiquetado 2553
— tipos de datos — 2553
Etiqueta 2553
— operaciones — 2554
TagResource (tag_resource) 2554
UntagResource (untag_resource) 2555
GetTags (get_tags) 2556
Tipos de datos comunes 2556
Etiqueta 2556
DecimalNumber 2557
ErrorDetail 2557
PropertyPredicate 2558
ResourceUri 2558
ColumnStatistics 2558
ColumnStatisticsError 2559
ColumnError 2559
ColumnStatisticsData 2560
BooleanColumnStatisticsData 2561
DateColumnStatisticsData 2561
DecimalColumnStatisticsData 2562
DoubleColumnStatisticsData 2562
LongColumnStatisticsData 2563

StringColumnStatisticsData 2563
BinaryColumnStatisticsData 2564
Patrones de cadena 2564
Excepciones 2566
AccessDeniedException 2566
AlreadyExistsException 2566
ConcurrentModificationException 2567
ConcurrentRunsExceededException 2567
CrawlerNotRunningException 2567
CrawlerRunningException 2567
CrawlerStoppingException 2568
EntityNotFoundException 2568
FederationSourceException 2568
FederationSourceRetryableException 2569
GlueEncryptionException 2569
IdempotentParameterMismatchException 2569
IllegalWorkflowStateException 2569
InternalServiceException 2570
InvalidExecutionEngineException 2570
InvalidInputException 2570
InvalidStateException 2570
InvalidTaskStatusTransitionException 2571
JobDefinitionErrorException 2571
JobRunInTerminalStateException 2571
JobRunInvalidStateTransitionException 2571
JobRunNotInTerminalStateException 2572
LateRunnerException 2572
NoScheduleException 2573
OperationTimeoutException 2573
ResourceNotReadyException 2573
ResourceNumberLimitExceededException 2573
SchedulerNotRunningException 2574
SchedulerRunningException 2574
SchedulerTransitioningException 2574
UnrecognizedRunnerException 2574
ValidationException 2575

VersionMismatchException ... 2575
AWS Glue Ejemplos de códigos de API ... 2576
Acciones ... 2584
CreateCrawler ... 2585
CreateJob ... 2597
DeleteCrawler ... 2609
DeleteDatabase ... 2615
DeleteJob ... 2620
DeleteTable ... 2627
GetCrawler ... 2632
GetDatabase ... 2641
GetDatabases ... 2650
GetJob ... 2653
GetJobRun ... 2655
GetJobRuns ... 2663
GetTables ... 2671
ListJobs ... 2684
StartCrawler ... 2690
StartJobRun ... 2700
Escenarios ... 2710
Comenzar a ejecutar rastreadores y trabajos ... 2710
Seguridad ... 2823
Protección de datos ... 2823
Cifrado en reposo ... 2824
Cifrado en tránsito ... 2842
Conformidad con FIPS ... 2843
Administración de claves ... 2843
Dependencia de AWS Glue en otros servicios de AWS ... 2844
Puntos de conexión de desarrollo ... 2844
Administración de identidades y accesos ... 2845
Público ... 2846
Autenticación con identidades ... 2847
Administración de acceso mediante políticas ... 2850
Cómo funciona AWS Glue con IAM ... 2853
Configuración de permisos de IAM para AWS Glue ... 2862
Ejemplos de políticas de control de acceso de AWS Glue ... 2895

AWS políticas gestionadas ... 2922
ARN del recurso ... 2930
Concesión de acceso entre cuentas ... 2937
Resolución de problemas ... 2945
Registro y monitoreo ... 2947
Validación de conformidad ... 2947
Resiliencia ... 2949
Seguridad de la infraestructura ... 2949
Puntos de conexión de VPC (AWS PrivateLink) ... 2950
VPC de Amazon compartidas ... 2952
Solución de problemas de AWS Glue ... 2954
Recopilación de información sobre la solución de problemas en AWS Glue ... 2954
Solución de errores de Spark ... 2955
Error: Resource unavailable (Recurso no disponible) ... 2956
Error: Could Not Find S3 Endpoint or NAT Gateway for subnetId in VPC (No se encontró el punto de conexión S3 o puerta de enlace NAT para subnetId en la VPC) ... 2957
Error: Inbound Rule in Security Group Required (La regla de entrada del grupo de seguridad es obligatoria) ... 2957
Error: Outbound Rule in Security Group Required (La regla de salida del grupo de seguridad es obligatoria) ... 2957
Error: la ejecución del trabajo ha fallado porque el rol transferido debe tener permisos de asuma el rol para el AWS Glue servicio ... 2958
Error: la DescribeVpcEndpoints acción no está autorizada. No se puede validar el ID de VPC vpc-id ... 2958
Error: la DescribeRouteTables acción no está autorizada. No se pudo validar el ID de subred: ID de subred en el ID de VPC: vpc-id ... 2958
Error: no se pudo llamar a ec2: DescribeSubnets ... 2958
Error: no se pudo llamar a ec2: DescribeSecurityGroups ... 2958
Error: Could Not Find Subnet for AZ (No se pudo encontrar la subred de AZ) ... 2959
Error: Job run exception when writing to a JDBC target (Excepción de ejecución de flujo de trabajo al escribir en un destino JDBC) ... 2959
Error: Amazon S3: The operation is not valid for the object's storage class ... 2960
Error: Amazon S3 timeout (Tiempo de espera de Amazon S3) ... 2960
Error: Amazon S3 access denied (Acceso denegado a Amazon S3) ... 2960
Error: Amazon S3 access key ID does not exist (El ID de clave de acceso de Amazon S3 no existe) ... 2960

Se produce un error en la ejecución de trabajo al obtener acceso a Amazon S3 con un URI de `s3a://` ... 2961
Error: Amazon S3 service token expired (Token de servicio de Amazon S3 caducado) ... 2963
Error: No private DNS for network interface found (No se encontró un DNS privado para la interfaz de red) ... 2963
Error: Development endpoint provisioning failed (Error en el aprovisionamiento del punto de conexión de desarrollo) ... 2963
Error: Notebook Server CREATE_FAILED (CREATE_FAILED el servidor del bloc de notas) ... 2964
Error: Local notebook fails to start (El bloc de notas local no se inicia) ... 2964
Error: Running crawler failed (Error de ejecución del rastreador) ... 2964
Error: Partitions were not updated (Las particiones no se han actualizado) ... 2965
Error: no se pudo actualizar el marcador de trabajo debido a que la versión no coincide ... 2965
Error: A job is reprocessing data when job bookmarks are enabled (Un trabajo está reprocesando datos cuando los marcadores de trabajo están habilitados) ... 2966
Error: comportamiento de conmutación por error entre VPC en AWS Glue ... 2967
Solucionar los errores del rastreador cuando éste utiliza las credenciales de Lake Formation ... 2968
Solución de errores de Ray ... 2971
Inspección de los registros de trabajos de Ray ... 2971
Solución de errores de trabajos de Ray ... 2972
Excepciones de machine learning en AWS Glue ... 2974
CancelMLTaskRunActivity ... 2974
CreateMLTaskRunActivity ... 2974
DeleteMLTransformActivity ... 2975
GetMLTaskRunActivity ... 2976
GetMLTaskRunsActivity ... 2976
GetMLTransformActivity ... 2976
GetMLTransformsActivity ... 2976
GetSaveLocationForTransformArtifactActivity ... 2977
GetTaskRunArtifactActivity ... 2977
PublishMLTransformModelActivity ... 2978
PullLatestMLTransformModelActivity ... 2978
PutJobMetadataForMLTransformActivity ... 2979
StartExportLabelsTaskRunActivity ... 2980
StartImportLabelsTaskRunActivity ... 2980

StartMLEvaluationTaskRunActivity 2981
StartMLLabelingSetGenerationTaskRunActivity 2982
UpdateMLTransformActivity 2982
Cuotas de AWS Glue 2983
Mejorar el AWS Glue rendimiento 2984
Ajuste de las estrategias para el tipo de trabajo 2984
Mejora del rendimiento de Spark 2984
Optimización de las lecturas con inserción 2985
Predica cómo insertar los archivos almacenados en Amazon S3 2985
Insertar al trabajar con fuentes JDBC 2987
Notas y limitaciones de cómo insertar en Glue AWS 2989
Utilizar Auto Scaling para AWS Glue 2990
Requisitos 2991
Habilitar Auto Scaling en AWS Glue Studio 1085
Habilitación de Auto Scaling con AWS CLI o SDK 1086
Monitorear Auto Scaling con las métricas de Amazon CloudWatch. 2993
Monitorear Auto Scaling con la IU de Spark 2994
Supervisión del uso de DPU en ejecución de trabajos de Auto Scaling 2995
Limitaciones 2995
Partición de cargas de trabajo con ejecución limitada 2995
Habilitar la partición de carga de trabajo 2996
Configuración de un desencadenador de AWS Glue para ejecutar el trabajo en forma automática 2997
Problemas conocidos 2999
Prevención del acceso a datos entre trabajos 2999
Historial de documentación 3002
Actualizaciones anteriores 3067
AWS Glosario 3069
..... mmmlxx

¿Qué es AWS Glue?

AWS Glue es un servicio de integración de datos sin servidor que facilita a los usuarios de análisis descubrir, preparar, migrar e integrar datos de varios orígenes. Puede utilizarlo para análisis, machine learning y desarrollo de aplicaciones. También incluye herramientas adicionales de productividad y operaciones de datos para la creación, la ejecución de trabajos y la implementación de flujos de trabajo empresariales.

Con AWS Glue, puede descubrir y conectarse a más de 70 orígenes de datos diversos y administrar sus datos en un catálogo de datos centralizado. Puede crear, ejecutar y supervisar visualmente canalizaciones de extracción, transformación y carga (ETL) para cargar datos en los lagos de datos. Además, puede buscar y consultar datos catalogados de forma inmediata mediante Amazon Athena, Amazon EMR y Amazon Redshift Spectrum.

AWS Glue combina las principales capacidades de integración de datos en un solo servicio. Estas capacidades incluyen el descubrimiento de datos, el ETL moderno, la limpieza, la transformación y la catalogación centralizada. Además, es un servicio sin servidor, lo que significa que no hay infraestructura para administrar. Con compatibilidad flexible para todas las cargas de trabajo como ETL, ELT y streaming en un solo servicio, AWS Glue admite usuarios en varias cargas de trabajo y varios tipos de usuarios.

Además, AWS Glue facilita la integración de datos en toda la arquitectura. Se integra con los servicios de análisis de AWS y los lagos de datos de Amazon S3. AWS Glue tiene interfaces de integración y herramientas de creación de trabajo que son fáciles de utilizar para todos los usuarios, desde desarrolladores hasta usuarios empresariales, con soluciones personalizadas para diversos conjuntos de habilidades técnicas.

Gracias a la capacidad de escalar bajo demanda, AWS Glue ayuda a centrarse en actividades de gran valor que maximizan el valor de los datos. Escala para cualquier tamaño de datos y admite todos los tipos de datos y las variaciones de esquema. Para aumentar la agilidad y optimizar los costos, AWS Glue brinda alta disponibilidad integrada y facturación de pago por uso.

Para obtener información sobre los precios, consulte Precios de AWS Glue.

AWS Glue Studio

AWS Glue Studio es una interfaz gráfica que facilita la creación, la ejecución y la supervisión de trabajos de integración de datos en AWS Glue. Puede componer visualmente flujos de trabajo de

transformación de datos y ejecutarlos sin problemas en el motor de ETL sin servidor basado en Apache Spark en AWS Glue.

Con AWS Glue Studio, puede crear y administrar trabajos que recopilan, transforman y limpian datos. También puede utilizar AWS Glue Studio para solucionar problemas y editar scripts de trabajo.

Temas

- Características de AWS Glue
- Conozca las innovaciones en AWS Glue
- Introducción a AWS Glue
- Acceso a AWS Glue
- Servicios relacionados

Características de AWS Glue

Las características de AWS Glue se dividen en tres categorías principales:

- Descubrimiento y organización de datos
- Transformación, preparación y limpieza de datos para análisis
- Creación y supervisión de canalizaciones de datos

Descubrimiento y organización de datos

- Unificación y búsqueda en varios almacenes de datos: almacene, indexe y busque en varios receptores y orígenes de datos mediante la catalogación de todos los datos en AWS.
- Descubrimiento automático de datos: utilice rastreadores de AWS Glue para inferir de forma automática la información del esquema e integrarla en AWS Glue Data Catalog.
- Administración de esquemas y permisos: valide y controle el acceso a las bases de datos y las tablas.
- Conexión a una amplia variedad de orígenes de datos: acceda a varios orígenes de datos, tanto en las instalaciones como en AWS, mediante las conexiones de AWS Glue para crear su lago de datos.

Transformación, preparación y limpieza de datos para análisis

- Transformaciones visuales de datos con una interfaz de lienzo de trabajo: defina el proceso de ETL en el editor de trabajos visual, y genere de forma automática el código para extraer, transformar y cargar los datos.
- Creación de canalizaciones de ETL complejas con programación de trabajo sencilla: invoque trabajos de AWS Glue según un horario, bajo demanda o en función de un evento.
- Limpieza y transformación de datos de streaming en tránsito: habilite el consumo continuo de datos, y límpielos y transfórmelos en tránsito. Esto hace que estén disponible para analizar en cuestión de segundos en el almacén de datos de destino.
- Deduplicación y limpieza de datos con machine learning integrado: limpie y prepare los datos para analizar sin convertirse en un experto en machine learning mediante el uso de la característica `FindMatches`. Esta característica deduplica y busca registros que son coincidencias imperfectas entre sí.
- Cuadernos de trabajo integrados: los cuadernos de trabajo de AWS Glue brindan cuadernos sin servidor con una configuración mínima de AWS Glue para que pueda comenzar a trabajar rápidamente.
- Edición, depuración y prueba del código de ETL: con las sesiones interactivas de AWS Glue, puede explorar y preparar datos de forma interactiva. Puede explorar, experimentar y procesar datos de forma interactiva con el IDE o el cuaderno que elija.
- Definición, detección y corrección de datos confidenciales: la detección de datos confidenciales de AWS Glue permite definir, identificar y procesar datos confidenciales en la canalización de datos y en el lago de datos.

Creación y supervisión de canalizaciones de datos

- Escalado automático según la carga de trabajo: escale y reduzca verticalmente y de forma dinámica los recursos en función de la carga de trabajo. Esto asigna trabajo a los trabajadores solo cuando es necesario.
- Automatización de trabajos con desencadenadores basados en eventos: inicie rastreadores o trabajos de AWS Glue con desencadenadores basados en eventos, y diseñe una cadena de trabajos y rastreadores dependientes.
- Ejecute y monitoree los trabajos: ejecute los trabajos AWS Glue con el motor que elija, Spark o Ray. Monitoreelos con herramientas de monitoreo automatizadas, información sobre la ejecución de los trabajos AWS Glue y AWS CloudTrail. Mejore el monitoreo de los trabajos respaldados por Spark con la interfaz de usuario de Apache Spark.

- Definición de flujos de trabajo para ETL y actividades de integración: defina los flujos de trabajo para ETL y las actividades de integración para varios rastreadores, trabajos y desencadenadores.

Conozca las innovaciones en AWS Glue

Conozca las últimas innovaciones AWS Glue y descubra cómo los clientes utilizan AWS Glue para habilitar la preparación de datos en forma de autoservicio en toda su organización.

Descubra cómo los clientes escalan AWS Glue más allá de la configuración tradicional y cómo configuran AWS Glue para el monitoreo y el rendimiento del trabajo.

Introducción a AWS Glue

Le recomendamos que lea las siguientes secciones:

- Información general del uso de AWS Glue
- Conceptos de AWS Glue
- Configuración de permisos de IAM para AWS Glue
- Introducción a AWS Glue Data Catalog
- Creación de trabajos en AWS Glue
- Introducción a las sesiones interactivas de AWS Glue
- Orquestación en AWS Glue

Acceso a AWS Glue

Puede crear, ver y administrar los trabajos de AWS Glue con cualquiera de las siguientes interfaces:

- Consola de AWS Glue: brinda una interfaz web para que pueda crear, ver y administrar los trabajos de AWS Glue. Para obtener acceso a la consola, consulte AWS Glue.
- AWS Glue Studio: brinda una interfaz gráfica para que pueda crear y editar visualmente los trabajos de AWS Glue. Para obtener más información, consulte ¿Qué es AWS Glue Studio?
- Sección AWS Glue de la referencia de la AWS CLI: brinda comandos de la AWS CLI que se pueden utilizar con AWS Glue. Para obtener más información, consulte Referencia de la AWS CLI para AWS Glue.

- API de AWS Glue: brinda una referencia de API completa para los desarrolladores. Para obtener más información, consulte API de AWS Glue.

Servicios relacionados

Los usuarios de AWS Glue también utilizan lo siguiente:

- AWS Lake Formation : un servicio que es una capa de autorización que brinda un control de acceso minucioso a los recursos en AWS Glue Data Catalog.
- AWS Glue DataBrew: una herramienta de preparación de datos visuales que se puede utilizar para limpiar y normalizar los datos sin tener que escribir ningún código.

AWS Glue: Cómo funciona

AWS Glue utiliza otros servicios de AWS para organizar los trabajos de ETL (extracción, transformación y carga) con el fin de crear almacenamientos de datos y lagos de datos, y generar flujos de salida. AWS Glue invoca operaciones de la API para transformar los datos, crear registros en tiempo de ejecución, almacenar su lógica de trabajo y crear notificaciones para monitorear las ejecuciones de trabajo. La consola de AWS Glue conecta estos servicios en una aplicación administrada, para que pueda concentrarse en la creación y monitorización de los trabajos de ETL. La consola desarrolla operaciones administrativas y de desarrollo de trabajos en su nombre. Puede proporcionar credenciales y otras propiedades a AWS Glue para obtener acceso a los orígenes de datos y escribir en los destinos de datos.

AWS Glue se encarga del aprovisionamiento y la administración de los recursos necesarios para ejecutar la carga de flujo de trabajo. No es necesario crear la infraestructura para una herramienta de ETL porque AWS Glue lo hace de forma automática. Cuando se necesitan recursos para reducir el tiempo de arranque, AWS Glue utiliza una instancia de su grupo de instancias para ejecutar la carga de flujo de trabajo.

Con AWS Glue, los trabajos se crean mediante las definiciones de tabla en el Data Catalog. Las trabajos se componen de scripts que incluyen la lógica de programación encargada de efectuar la transformación. Puede utilizar disparadores para iniciar trabajos en función de una programación o a través del resultado de un evento especificado. Puede determinar dónde van a residir los datos y qué datos de origen van a rellenar el destino. Con la entrada, AWS Glue genera el código necesario para transformar los datos desde el origen hasta el destino. También puede proporcionar scripts en la consola de AWS Glue o la API para procesar los datos.

Orígenes y destinos de datos

AWS Glue permite leer y escribir datos de varios sistemas y bases de datos, entre los que se incluyen:

- Amazon S3
- Amazon DynamoDB
- Amazon Redshift
- Amazon Relational Database Service (Amazon RDS)
- Bases de datos accesibles a JDBC de terceros

- MongoDB y Amazon DocumentDB (compatible con MongoDB)
- Otros conectores de Marketplace y complementos de Apache Spark

Flujos de datos

AWS Glue para Spark puede transmitir datos desde los siguientes sistemas:

- Amazon Kinesis Data Streams
- Apache Kafka

AWS Glue está disponible en distintas regiones de AWS. Para obtener más información, consulte Regiones y puntos de conexión de AWS en la Referencia general de Amazon Web Services.

Temas

- Los trabajos de ETL sin servidor se ejecutan de forma aislada
- Conceptos de AWS Glue
- Componentes de AWS Glue
- AWS Glue para Spark y AWS Glue para Ray
- Conversión de esquemas semiestructurados a esquemas relacionales con AWS Glue
- AWS Sistemas tipo Glue

Los trabajos de ETL sin servidor se ejecutan de forma aislada

AWS Glue ejecuta los trabajos de ETL en un entorno sin servidor, con el motor elegido, Spark o Ray. AWS Glue ejecuta estos trabajos en recursos virtuales que aprovisiona y gestiona en su propia cuenta de servicio.

AWS Glue se ha diseñado para poder:

- Segregar datos de clientes.
- Proteger los datos de clientes en tránsito y en reposo.
- Obtenga acceso a los datos de clientes únicamente según se necesite para dar respuesta a las solicitudes de los clientes, mediante credenciales provisionales y con permisos reducidos o con el consentimiento del cliente para roles de IAM en su cuenta.

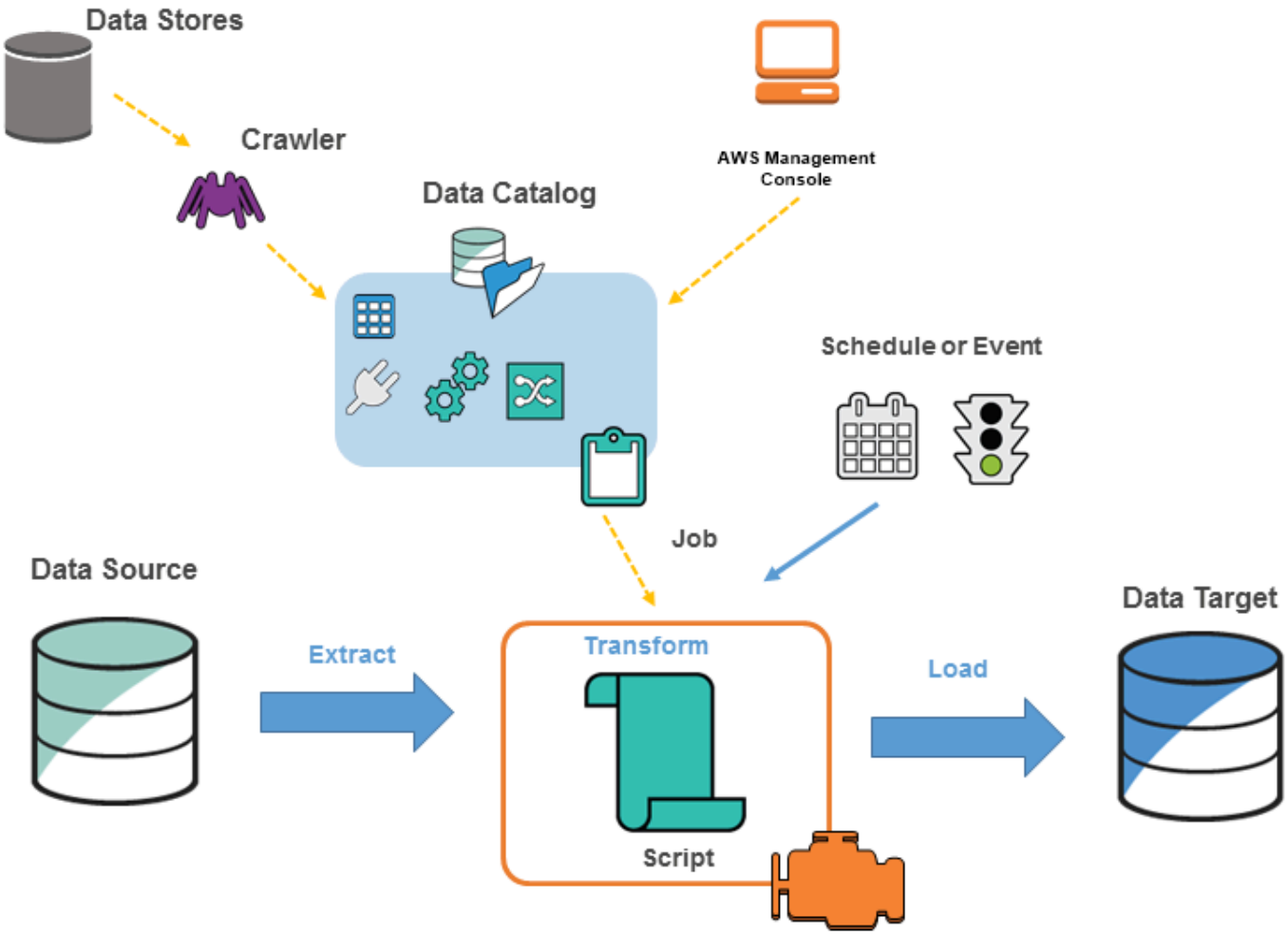
Durante el aprovisionamiento de un flujo de trabajo de ETL, usted proporciona los orígenes de datos de entrada y los destinos de datos de salida en la nube virtual privada (VPC). Además, debe proporcionar el rol de IAM, el ID de la VPC, el ID de la subred y el grupo de seguridad; todos ellos necesarios para obtener acceso a los orígenes y los destinos de datos. Para cada una de las tuplas (el ID de la cuenta de cliente, el rol de IAM, el ID de subred y el grupo de seguridad), AWS Glue crea un nuevo entorno que se aísla en la red y en las tareas de administración de todos los entornos que estén en la cuenta de servicio de AWS Glue.

AWS Glue crea interfaces de red elástica en la subred mediante direcciones IP privadas. Los trabajos utilizan estas interfaces de red elástica para obtener acceso a los orígenes de datos y los destinos de datos. El tráfico entrante y saliente del entorno de ejecución del trabajo, así como el interno, está regido por las políticas de VPC y de red, con una única excepción: las llamadas realizadas a las bibliotecas de AWS Glue pueden utilizar un proxy para el tráfico de las operaciones de la API de AWS Glue a través de la VPC de AWS Glue. Se registran todas las llamadas al API de AWS Glue, por tanto, los propietarios de datos pueden auditar el acceso a la API mediante la habilitación de AWS CloudTrail, que entrega registros de auditoría a su cuenta.

Los entornos que administra AWS Glue y que ejecutan los trabajos de ETL están protegidos a través de las mismas prácticas de seguridad que aplican otros servicios de AWS. Para obtener información general de las prácticas y las responsabilidades de seguridad compartidas, consulte el documento técnico de Introducción a la seguridad de procesos de AWS.

Conceptos de AWS Glue

En el siguiente diagrama se muestra la arquitectura de un entorno de AWS Glue.



Los trabajos se definen en AWS Glue a fin de realizar el flujo de trabajo necesario para extraer, transformar y cargar datos (ETL) desde un origen de datos hasta un destino de datos. Normalmente, usted llevará a cabo las siguientes acciones:

- Para los orígenes de almacén de datos, puede definir un rastreador para rellenar su AWS Glue Data Catalog con definiciones de tabla de metadatos. Puede dirigir su rastreador a un almacén de datos y el rastreador crea definiciones de tabla en el Data Catalog. Para orígenes de streaming, se definen manualmente tablas del Data Catalog y se especifican las propiedades del flujo de datos.

  Además de las definiciones de tabla, el AWS Glue Data Catalog contiene otros metadatos necesarios para definir los trabajos de ETL. Utilizará estos metadatos al definir un flujo de trabajo para transformar sus datos.

- AWS Glue puede generar un script para transformar sus datos. O bien, puede proporcionar el script en la consola o API de AWS Glue.
- Puede ejecutar su flujo de trabajo bajo demanda o configurarlo de modo que se inicie al activarse un disparador especificado. El disparador puede corresponder a una programación basada en tiempo o a un evento.

  Al ejecutarse su flujo de trabajo, un script extrae datos de su origen de datos, transforma los datos y los carga en su destino de datos. El script se ejecuta en un entorno Apache Spark en AWS Glue.

> **Important**
>
> Las tablas y las bases de datos de AWS Glue son objetos en el AWS Glue Data Catalog. Contienen metadatos; no datos de un almacén de datos.

Los datos basados en texto, como los CSV, deben estar codificados en **`UTF-8`** para que AWS Glue los procese de forma exitosa. Para obtener más información, consulte UTF-8 en Wikipedia.

Terminología de AWS Glue

AWS Glue se basa en la interacción de varios componentes para crear y gestionar su flujo de trabajo de extracción, transformación y carga (ETL).

AWS Glue Data Catalog

El almacén de metadatos persistentes en AWS Glue. Contiene definiciones de tablas, definiciones de trabajos y otra información de control para administrar su entorno de AWS Glue. Cada cuenta de AWS tiene un AWS Glue Data Catalog por región.

Clasificador

Determina el esquema de sus datos. AWS Glue proporciona clasificadores para tipos de archivos comunes, como CSV, JSON, AVRO, XML y otros. También proporciona clasificadores para sistemas de administración de bases de datos relacionales comunes mediante una conexión de JDBC. Puede escribir su propio clasificador mediante un patrón de grok o especificando una etiqueta de fila en un documento XML.

Connection

Un objeto del Data Catalog que contiene las propiedades necesarias para conectarse a un almacén de datos determinado.

Rastreador

Un programa que se conecta a un almacén de datos (origen o destino), avanza por una lista de prioridades de clasificadores para determinar el esquema de sus datos y, a continuación, crea tablas de metadatos en el AWS Glue Data Catalog.

Base de datos

Un conjunto de definiciones de tabla del Data Catalog asociadas, organizadas en un grupo lógico.

Almacén de datos, origen de datos, destino de datos

Un almacén de datos es un repositorio para almacenar los datos de forma persistente. Entre los ejemplos se incluyen buckets de Amazon S3 y bases de datos relacionales. Un origen de datos es un almacén de datos que se utiliza como entrada para un proceso o una transformación. Un destino de datos es un almacén de datos en el que escribe un proceso o una transformación.

Punto de enlace de desarrollo

Un entorno que puede utilizar para desarrollar y probar los scripts ETL de AWS Glue.

Marco dinámico

Tabla distribuida que admite datos anidados como estructuras y matrices. Cada registro se autodescribe y está diseñado para flexibilidad de esquemas con datos semiestructurados. Cada registro contiene tanto los datos como el esquema que describe esos datos. Puede usar marcos dinámicos y Apache Spark DataFrames en sus scripts de ETL, y realizar conversiones entre ellos. Las tramas dinámicas proporcionan un conjunto de transformaciones avanzadas para la limpieza de datos y ETL.

Trabajo

La lógica empresarial que es necesaria para realizar el flujo de trabajo de ETL. Se compone de un script de transformación, orígenes de datos y destinos de datos. Las ejecuciones de trabajos pueden iniciarse a partir de disparadores programados o activados por eventos.

Panel de rendimiento del trabajo

AWS Glue proporciona un panel de ejecución integral para sus trabajos de ETL. El panel muestra información sobre las ejecuciones de trabajos desde un periodo específico.

Interfaz de cuaderno

Una experiencia de cuaderno mejorada con una configuración con un solo clic para facilitar la creación de trabajos y la exploración de datos. El bloc de notas y las conexiones se configuran automáticamente. Puede utilizar la interfaz de cuaderno basada en Jupyter Notebook para desarrollar, depurar e implementar scripts y flujos de trabajo mediante infraestructura de ETL de Apache Spark sin servidor de AWS Glue. También puede realizar consultas ad hoc, análisis de datos y visualización (por ejemplo, tablas y gráficos) en el entorno de cuaderno.

Script

Código que extrae datos de orígenes, los transforma y los carga en destinos. AWS Glue genera scripts PySpark o Scala.

Tabla

La definición de metadatos que representa sus datos. Independientemente de si sus datos están en un archivo de Amazon Simple Storage Service (Amazon S3), una tabla de Amazon Relational Database Service (Amazon RDS) u otro conjunto de datos, la tabla define el esquema de sus datos. Una tabla de AWS Glue Data Catalog está formada por los nombres de las columnas, las definiciones de tipos de datos, la información de partición y otros metadatos acerca de un conjunto de datos base. El esquema de sus datos viene representado en su definición de tabla de AWS Glue. Los datos reales permanecen en su almacén de datos original, ya sea en un archivo o en una tabla de base de datos relacional. AWS Glue cataloga sus archivos y tablas de bases de datos relacionales en el AWS Glue Data Catalog. Estos se usan como orígenes y destinos al crear un flujo de trabajo de ETL.

Transformar

La lógica de código que se usa para manipular sus datos en un formato diferente.

Desencadenador

Inicia un flujo de trabajo de ETL. Los disparadores se pueden definir según un momento programado o un evento.

Editor visual de trabajos

El editor de trabajos visuales es una interfaz gráfica que facilita la creación, ejecución y supervisión de los trabajos de extracción, transformación y carga (ETL) en AWS Glue. Puede componer visualmente flujos de trabajo de transformación de datos y ejecutarlos sin problemas en el motor de ETL sin servidor basado en Apache Spark de AWS Glue e inspeccionar el esquema y los datos resultantes en cada paso del trabajo.

Entorno de trabajo

Con AWS Glue, solo paga por el tiempo que tarda en ejecutarse su trabajo de ETL. No hay que administrar recursos ni hay costos iniciales. No se le cobra por el tiempo de inicio o cierre. Se le cobra una tarifa por hora basada en el número de unidades de procesamiento de datos (o DPU) utilizadas para ejecutar el trabajo de ETL. Una sola unidad de procesamiento de datos (DPU) también se denomina empleado. AWS Glue cuenta con tres tipos de empleados para ayudarlo a seleccionar la configuración que cumpla con sus requisitos de latencia y costo del trabajo. Los procesos de trabajo pueden tener configuraciones estándar, G.1X, G.2X y G.025X.

Componentes de AWS Glue

AWS Glue proporciona una consola y operaciones de API para configurar y administrar su carga de flujo de trabajo de extracción, transformación y carga (ETL). Puede usar las operaciones de API a través de varios SDK específicos de lenguaje y la AWS Command Line Interface (AWS CLI). Para obtener información sobre el uso de AWS CLI, consulte Referencias de comandos de AWS CLI.

AWS Glue usa el AWS Glue Data Catalog para almacenar metadatos acerca de orígenes de datos, transformaciones y destinos. El Data Catalog es un reemplazo instantáneo para el Apache Hive Metastore. AWS Glue Jobs system proporciona una infraestructura administrada para definir, programar y ejecutar operaciones de ETL en sus datos. Para obtener más información sobre la API de AWS Glue, consulte AWS Glue API.

Consola de AWS Glue

Use la consola de AWS Glue para definir y orquestar su flujo de flujo de trabajo de ETL. La consola llama a varias operaciones API en el AWS Glue Data Catalog y AWS Glue Jobs system para realizar las siguientes tareas:

- Definir objetos de AWS Glue como trabajos, tablas, rastreadores y conexiones.

- Programar cuándo se ejecutan los rastreadores.
- Definir eventos o programaciones para los disparadores de trabajos.
- Buscar y filtrar listas de objetos de AWS Glue.
- Editar scripts de transformación.

AWS Glue Data Catalog

El AWS Glue Data Catalog es su almacén de metadatos técnicos persistente en la nube de AWS.

Cada cuenta de AWS tiene un AWS Glue Data Catalog por región de AWS. Cada catálogo de datos es una colección de tablas altamente escalable organizadas en bases de datos. Una tabla es la representación de metadatos de una colección de datos estructurados o semiestructurados almacenados en orígenes como Amazon RDS, Apache Hadoop Distributed File System, Amazon OpenSearch Service y otros. AWS Glue Data Catalog proporciona un repositorio uniforme donde sistemas dispares pueden almacenar y encontrar metadatos para hacer un seguimiento de los datos en silos de datos. A continuación, puede utilizar los metadatos para consultar y transformar esos datos de forma coherente en una amplia variedad de aplicaciones.

Utilice el catálogo de datos junto con políticas de AWS Identity and Access Management y Lake Formation para controlar el acceso a las tablas y bases de datos. Al hacer esto, permite a los diversos grupos de su empresa publicar datos de forma segura en toda la organización, al mismo tiempo que se protege la información confidencial de forma altamente granular.

El catálogo de datos, junto con CloudTrail y Lake Formation, también proporciona capacidades de auditoría y gobernanza completas, con seguimiento de cambios de esquema y controles de acceso de datos. Esto ayuda a garantizar que los datos no se modificaron incorrectamente o no se compartieron sin querer.

Para obtener información sobre cómo proteger y auditar el AWS Glue Data Catalog, consulte:

- AWS Lake Formation: para obtener más información, consulte ¿Qué es AWS Lake Formation? en la Guía para desarrolladores de AWS Lake Formation.
- CloudTrail: para obtener más información, consulte What Is CloudTrail? (¿Qué es CloudTrail?) en la Guía del usuario de AWS CloudTrail.

Los siguientes son otros servicios y proyectos de código abierto de AWS que utilizan el AWS Glue Data Catalog:

- Amazon Athena: para obtener más información, consulte Descripción de las tablas, bases de datos y el catálogo de datos en la Guía del usuario de Amazon Athena.
- Amazon Redshift Spectrum: para obtener más información, consulte Using Amazon Redshift Spectrum to Query External Data (Uso de Amazon Redshift Spectrum para consultar datos externos) en la Guía para desarrolladores de bases de datos Amazon Redshift.
- Amazon EMR: para obtener más información, consulte Resource-Based Policies for Amazon EMR Access to AWS Glue Data Catalog (Uso de políticas basadas en recursos para acceso de Amazon EMR al) en la Guía de administración de Amazon EMR.
- Cliente de AWS Glue Data Catalog para el almacén de metadatos de Apache Hive: para obtener más información sobre este proyecto de GitHub, consulte Cliente de AWS Glue Data Catalog para el almacén de metadatos de Apache Hive.

Rastreadores y clasificadores de AWS Glue

AWS Glue también le permite configurar rastreadores que pueden analizar datos en toda clase de repositorios, clasificarlos, extraer información de esquema de ellos y almacenar los metadatos de forma automática en el AWS Glue Data Catalog. El AWS Glue Data Catalog se puede usar para guiar las operaciones de ETL.

Para obtener información acerca de cómo configurar rastreadores y clasificadores, consulte Uso de rastreadores para completar el Catálogo de datos . Para obtener información acerca de cómo programar rastreadores y clasificadores mediante la API de AWS Glue, consulte API de rastreadores y clasificadores.

Operaciones de ETL de AWS Glue

Al usar los metadatos en el Data Catalog, AWS Glue puede generar automáticamente scripts de Scala o PySpark (la API de Python para Apache Spark) con extensiones de AWS Glue que puede usar y modificar para realizar diversas operaciones de ETL. Por ejemplo, puede extraer, limpiar y transformar datos sin formato y, a continuación, almacenar el resultado en un repositorio distinto, donde se puede consultar y analizar. Dicho script puede convertir un archivo CSV en un formato relacional y guardarlo en Amazon Redshift.

Para obtener más información acerca de cómo usar capacidades de ETL de AWS Glue, consulte Programación de scripts de Spark.

ETL de streaming en AWS Glue

AWS Glue le permite realizar operaciones de ETL en datos de streaming mediante trabajos en ejecución continua. ETL de streaming de AWS Glue se basa en el motor Apache Spark Structured Streaming, y puede capturar flujos de Amazon Kinesis Data Streams, Apache Kafka y Amazon Managed Streaming for Apache Kafka (Amazon MSK). ETL de streaming puede limpiar y transformar los datos de streaming y cargarlos en almacenes de datos de Amazon S3 o JDBC. Utilice ETL de streaming en AWS Glue para procesar datos de eventos como transmisiones de IoT, transmisiones de clics y registros de red.

Si conoce el esquema del origen de datos de streaming, puede especificarlo en una tabla del Data Catalog. De lo contrario, puede habilitar la detección de esquemas en el trabajo de ETL de streaming. El trabajo determinará en forma automática el esquema a partir de los datos entrantes.

El script de ETL puede usar las transformaciones incorporadas de AWS Glue y las transformaciones nativas de Apache Spark Structured Streaming. Para obtener más información, consulte Operaciones en DataFrames/conjuntos de datos de streaming en el sitio web de Apache Spark.

Para obtener más información, consulte the section called “Trabajos ETL de streaming”.

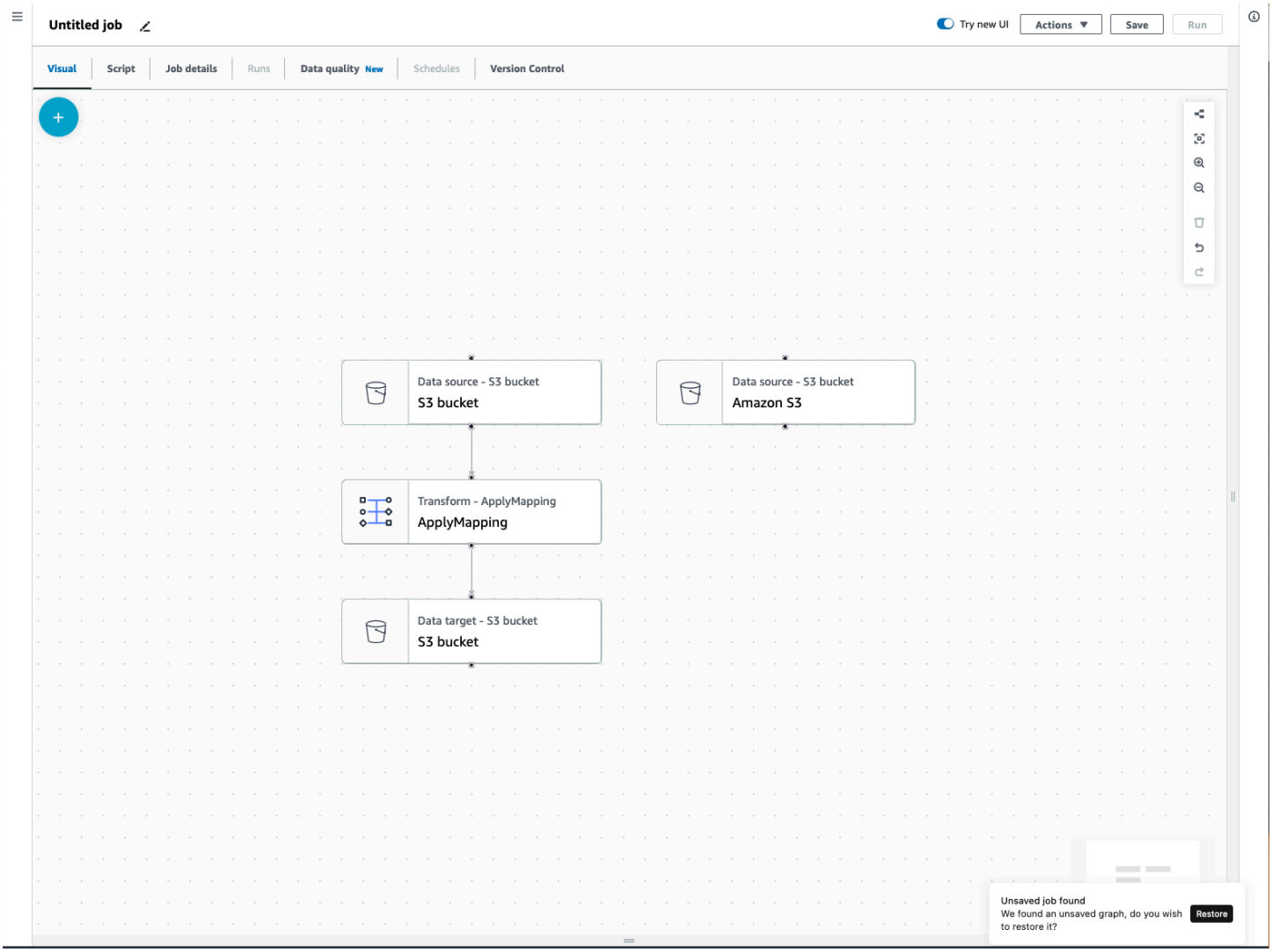
El sistema de trabajos de AWS Glue

AWS Glue Jobs system proporciona infraestructura administrada para orquestar su flujo de flujo de trabajo de ETL. Puede crear trabajos en AWS Glue que automaticen los scripts que usa para extraer, transformar y transferir datos a distintas ubicaciones. Los trabajos se pueden programar y encadenar, o bien eventos como la llegada de nuevos datos pueden activarlos.

Para obtener más información acerca del uso de AWS Glue Jobs system, consulte Supervisión de AWS Glue. Para obtener información acerca de la programación del uso de la API de AWS Glue Jobs system, consulte API de trabajos.

Componentes de ETL visuales

AWS Glue permite crear trabajos de ETL a través de un lienzo visual que pueda manipular.



Menú de trabajos de ETL

Las opciones del menú en la parte superior del lienzo permiten acceder a las distintas vistas y detalles de configuración del trabajo.

- Visual: el lienzo del editor de trabajos de Visual. Aquí es donde puede agregar nodos para crear un trabajo.
- Guion: la representación en guion de su trabajo de ETL. AWS Glue genera el guion en función de la representación visual del trabajo. También puede editar el guion o descargarlo.

> **Note**
>
> Si decide editar el guion, la experiencia de creación de trabajos se convierte permanentemente en un modo de solo guion. Después ya no podrá utilizar el editor visual

para editar el trabajo. Debe agregar todos los orígenes, transformaciones y destinos del trabajo y realizar todos los cambios que necesite con el editor visual antes de decidir editar el guion.

- Detalles del trabajo: la pestaña Detalles del trabajo permite configurar el trabajo mediante la configuración de las propiedades del trabajo. Hay propiedades básicas, como el nombre y la descripción del trabajo, el rol de IAM, el tipo de trabajo, la versión de AWS Glue, el idioma, el tipo de trabajador, el número de trabajadores, el marcador del trabajo, la ejecución flexible, el número de retiros y el tiempo de espera del trabajo. También hay propiedades avanzadas, como las conexiones, las bibliotecas, los parámetros del trabajo y las etiquetas.
- Ejecuciones: después de ejecutar el trabajo, se puede acceder a esta pestaña para ver las ejecuciones anteriores.
- Calidad de datos: calidad de datos evalúa y supervisa la calidad de sus activos de datos. Puede obtener más información sobre cómo utilizar la calidad de los datos en esta pestaña y agregar una transformación de la calidad de los datos al trabajo.
- Programas: Los trabajos que ha programado aparecen en esta pestaña. Si no hay ningún programa adjunto a este trabajo, no se puede acceder a esta pestaña.
- Control de versiones: puede usar Git con el trabajo al configurarlo en un repositorio de Git.

Paneles de ETL visuales

Cuando trabaja en el lienzo, hay varios paneles disponibles para ayudarle a configurar los nodos o a obtener una vista previa de los datos y ver el esquema de salida.

- Propiedades: el panel Propiedades aparece al elegir un nodo del lienzo.
- Vista previa de datos: el panel de vista previa de datos proporciona una vista preliminar de la salida de datos para que pueda tomar decisiones antes de ejecutar el trabajo y examinar la salida.
- Esquema de salida: la pestaña Esquema de salida permite ver y editar el esquema de los nodos de transformación.

Redimensionar los paneles

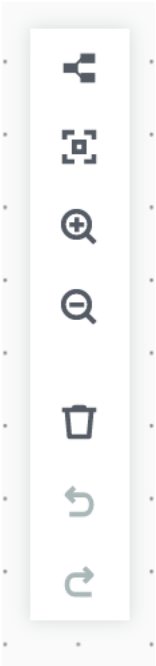
Puede cambiar el tamaño del panel de Propiedades situado en el lado derecho de la pantalla y en el panel inferior, que contiene las pestañas de Vista previa de datos y Esquema de salida al hacer clic en el borde del panel y arrastrarlo hacia la izquierda y hacia la derecha o hacia arriba y hacia abajo.

- Panel de propiedades: cambie el tamaño del panel de propiedades al hacer clic y arrastrar el borde del lienzo situado en el lado derecho de la pantalla y arrastrándolo hacia la izquierda para ampliar el ancho. De forma predeterminada, el panel está contraído y, cuando se selecciona un nodo, el panel de propiedades se abre con un tamaño predeterminado.
- Panel de vista previa de datos y esquema de salida: cambie el tamaño del panel inferior al hacer clic y arrastrar el borde inferior del lienzo en la parte inferior de la pantalla. Luego arrástrelo hacia arriba para ampliar la altura. De forma predeterminada, el panel está contraído y, cuando se selecciona un nodo, el panel inferior se abre con un tamaño predeterminado.

Lienzo de trabajo

Puede agregar, eliminar y mover o reordenar nodos directamente en el lienzo de ETL visuales. Considérelo su espacio de trabajo para crear un trabajo de ETL completamente funcional que comience con un origen de datos y termine con un destino de datos.

Cuando trabaja con nodos en el lienzo, cuenta con una barra de herramientas que puede ayudarle a acercar y alejar la imagen, eliminar nodos, establecer o editar conexiones entre nodos, cambiar la orientación del flujo de trabajo y deshacer o rehacer una acción.



La barra de herramientas flotante está anclada al tamaño superior derecho del lienzo y contiene varias imágenes que realizan acciones:

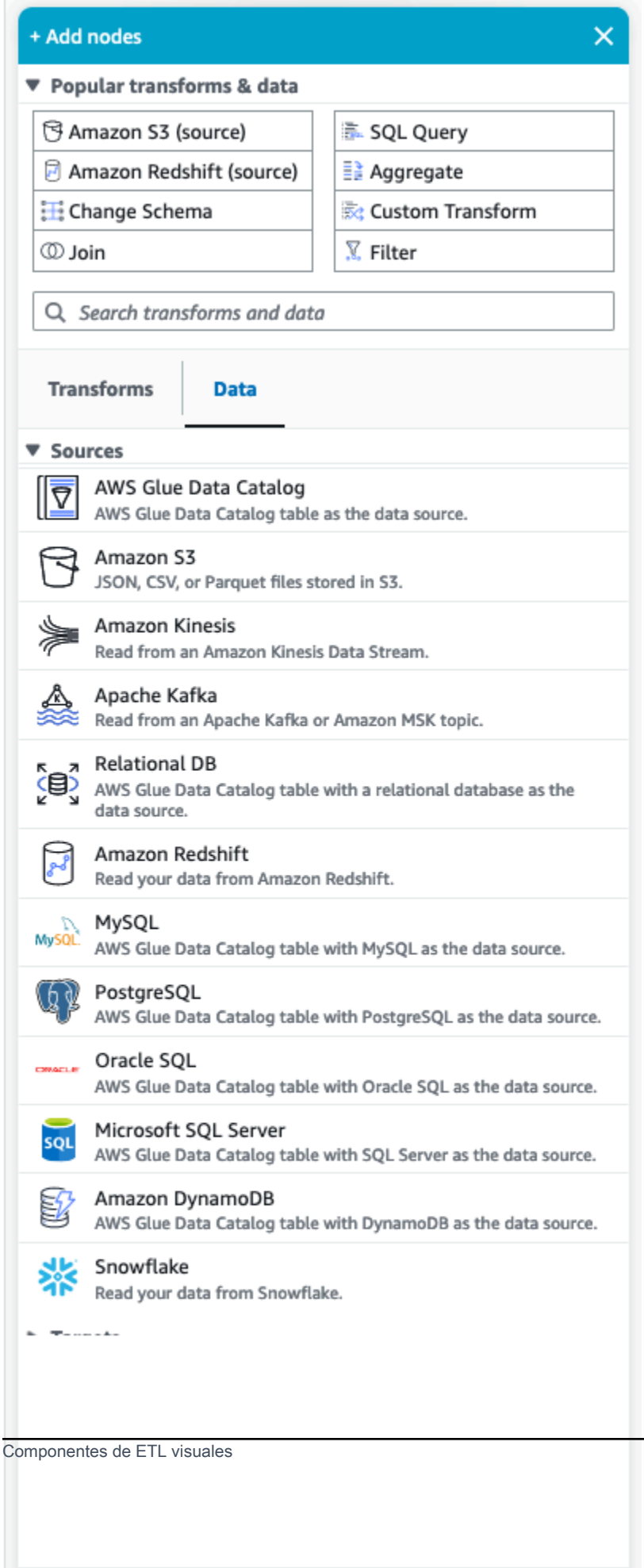
- Icono de diseño: el primer icono de la barra de herramientas es el icono de diseño. La dirección de las tareas visuales es de arriba a abajo de forma predeterminada. Reorganiza la dirección de las tareas visuales al organizar los nodos horizontalmente de izquierda a derecha. Al volver a hacer clic en el icono de diseño, se cambia la dirección de arriba a abajo.
- Icono de recentrar: el icono de recentrar cambia la vista del lienzo al centrarlo. Puede usarlo con trabajos grandes para volver a la posición central.
- Ícono de zoom: el icono de zoom amplía el tamaño de los nodos del lienzo.
- Ícono de alejar: el icono de zoom achica el tamaño de los nodos del lienzo.
- Icono de papelera: el icono de papelera elimina un nodo del trabajo visual. Primero debe seleccionar un nodo.
- Icono de deshacer: el icono de deshacer invierte la última acción realizada en el trabajo visual.
- Icono de rehacer: el icono de rehacer repite la última acción realizada en el trabajo visual.

Uso del minimapa

Panel de recursos

El panel de recursos contiene todos los orígenes de datos, las acciones de transformación y las conexiones disponibles. Abre el panel de recursos en el lienzo mediante un clic en el icono “+”. Esto abrirá el panel de recursos.

Para cerrar el panel de recursos, haz clic en la X situada en la esquina superior derecha del panel de recursos. Esto ocultará el panel hasta que estés listo para volver a abrirlo.


+ Add nodes
Popular transforms & data
Amazon S3 (source)
Amazon Redshift (source)
Change Schema
Join
SQL Query
Aggregate
Custom Transform
Filter
Search transforms and data
Transforms
Data
Sources
AWS Glue Data Catalog
AWS Glue Data Catalog table as the data source.
Amazon S3
JSON, CSV, or Parquet files stored in S3.
Amazon Kinesis
Read from an Amazon Kinesis Data Stream.
Apache Kafka
Read from an Apache Kafka or Amazon MSK topic.
Relational DB
AWS Glue Data Catalog table with a relational database as the data source.
Amazon Redshift
Read your data from Amazon Redshift.
MySQL
AWS Glue Data Catalog table with MySQL as the data source.
PostgreSQL
AWS Glue Data Catalog table with PostgreSQL as the data source.
Oracle SQL
AWS Glue Data Catalog table with Oracle SQL as the data source.
Microsoft SQL Server
AWS Glue Data Catalog table with SQL Server as the data source.
Amazon DynamoDB
AWS Glue Data Catalog table with DynamoDB as the data source.
Snowflake
Read your data from Snowflake.

Transformaciones populares y datos

En la parte superior del panel hay una colección de transformaciones populares y datos. Estos nodos se utilizan habitualmente en AWS Glue. Elija uno para agregarlo al lienzo. También puede ocultar las Transformaciones populares y datos al hacer clic en el triángulo situado junto al encabezado Transformaciones populares y datos.

En la sección Transformaciones populares y datos, puede buscar transformaciones y nodos de orígenes de datos. Los resultados aparecen a medida que escribe. Cuantas más letras agregue a la búsqueda, la lista de resultados se reducirá. Los resultados de la búsqueda se llenan a partir del nombre o la descripción del nodo. Elija el nodo para agregarlo al lienzo.

Transformaciones y datos

Hay dos pestañas que organizan los nodos en transformaciones y datos.

Transformaciones: al elegir la pestaña Transformaciones, se pueden seleccionar todas las transformaciones disponibles. Elija una transformación para agregarla al lienzo. También puede elegir Agregar transformación en la parte inferior de la lista de transformaciones, lo que abrirá una nueva página con la documentación para crear transformaciones visuales personalizadas. Si sigue estos pasos, podrá crear sus propias transformaciones. Sus transformaciones aparecerán en la lista de transformaciones disponibles.

Datos: la pestaña de datos contiene todos los nodos de los orígenes y los destinos. Puede ocultar los orígenes y los destinos al hacer clic en el triángulo situado junto al encabezado Orígenes o destinos. Puede mostrar los orígenes y los destinos al hacer clic de nuevo en el triángulo. Elija un nodo de origen o de destino para agregarlo al lienzo. También puede elegir Administrar conexiones para agregar una nueva conexión. Esto abrirá la página de conectores en la consola.

AWS Glue para Spark y AWS Glue para Ray

En AWS Glue en Apache Spark (ETL de AWS Glue), puede utilizar PySpark para escribir código Python para gestionar datos a escala. Spark es una solución conocida para este problema, pero a los ingenieros de datos con experiencia en Python puede parecerles poco intuitiva la transición. El modelo de DataFrame de Spark no es perfectamente como el de Python, lo que refleja el lenguaje de Scala y el entorno de ejecución de Java en el que se basa.

En AWS Glue, puede utilizar los trabajos del intérprete de comandos de Python para ejecutar integraciones de datos nativas de Python. Estos trabajos se ejecutan en una única instancia de

Amazon EC2 y se ven limitados por la capacidad de esa instancia. Esto restringe el rendimiento de los datos que puede procesar y resulta costoso mantenerlos cuando se trata de macrodatos.

AWS Glue para Ray permite escalar verticalmente las cargas de trabajo de Python sin una inversión sustancial en el aprendizaje de Spark. Puede aprovechar ciertos escenarios en los que Ray tiene un mejor rendimiento. Al ofrecerle una opción, puede utilizar los puntos fuertes de Spark y Ray.

ETL de AWS Glue y AWS Glue para Ray son diferentes en el fondo, por lo que admiten diferentes características. Consulte la documentación para ver las características admitidas.

¿Qué es AWS Glue para Ray?

Ray es un marco de computación distribuida de código abierto que se puede utilizar para escalar verticalmente las cargas de trabajo, con un enfoque en Python. Para obtener información sobre Ray, consulte el sitio web de Ray. AWS Glue Los trabajos y las sesiones interactivas de Ray le permiten utilizar Ray en AWS Glue.

Puede usar AWS Glue para Ray para escribir scripts de Python para cálculos que se ejecutarán en paralelo en varias máquinas. En los trabajos y sesiones interactivas de Ray, puede utilizar bibliotecas de Python conocidas, como pandas, para facilitar la escritura y la ejecución de los flujos de trabajo. Para obtener más información sobre los conjuntos de datos de Ray, consulte Conjuntos de datos de Ray en la documentación de Ray. Para más información acerca de Pandas, consulte el sitio web de Pandas.

Cuando utiliza AWS Glue para Ray, puede ejecutar los flujos de trabajo de pandas con macrodatos a escala empresarial, con solo algunas líneas de código. Puede crear un informe desde la consola de AWS Glue o AWS SDK. También puede abrir una sesión interactiva de AWS Glue para ejecutar el código en un entorno de Ray sin servidor. Aún no se admiten los trabajos visuales en AWS Glue Studio.

Los trabajos de AWS Glue para Ray permiten ejecutar un script según un cronograma o en respuesta a un evento de Amazon EventBridge. Los trabajos almacenan información de registro y estadísticas de supervisión en CloudWatch que permiten comprender el estado y la fiabilidad del script. Para más información sobre el sistema de trabajos de AWS Glue, consulte the section called “Trabajar con tareas de Ray”.

Las sesiones interactivas de AWS Glue para Ray (vista previa) permiten ejecutar fragmentos de código uno tras otro en los mismos recursos aprovisionados. Puede usarlo para crear prototipos y desarrollar scripts de manera eficiente, o bien crear sus propias aplicaciones interactivas. Puede

utilizar las sesiones interactivas de AWS Glue desde los cuadernos de AWS Glue Studio en AWS Management Console. Para obtener más información, consulte Uso de los cuadernos con AWS Glue Studio y AWS Glue. También puede utilizarlos mediante un kernel de Jupyter que permite ejecutar sesiones interactivas desde las herramientas de edición de código existentes compatibles con cuadernos de Jupyter, como VSCode. Para obtener más información, consulte the section called "AWS Glue para las sesiones interactivas de Ray (versión preliminar)".

Ray automatiza el trabajo de escalar el código de Python mediante la distribución del procesamiento en un clúster de equipos que vuelve a configurar en tiempo real en función de la carga. Esto puede mejorar el rendimiento por dólar de determinadas cargas de trabajo. Con los trabajos de Ray, hemos incorporado el escalado automático de forma nativa en el modelo de trabajo de AWS Glue, para poder aprovechar al máximo esta característica. Los trabajos de Ray se ejecutan en AWS Graviton, lo que lleva a una mayor rentabilidad general de los precios.

Además de ahorrar costos, puede utilizar el escalado automático nativo para ejecutar cargas de trabajo de Ray sin invertir tiempo en el mantenimiento, el ajuste y la administración del clúster. Puede utilizar bibliotecas conocidas de código abierto listas para su uso, como pandas y AWS SDK para Pandas. Estas mejoran la velocidad de iteración mientras desarrolla en AWS Glue para Ray. Cuando utilice AWS Glue para Ray, podrá desarrollar y ejecutar rápidamente cargas de trabajo rentables de integración de datos.

Conversión de esquemas semiestructurados a esquemas relacionales con AWS Glue

Es habitual querer convertir los datos semiestructurados en tablas relacionales. Conceptualmente, usted aplana un esquema jerárquico a un esquema relacional. AWS Glue puede realizar esta conversión automáticamente sobre la marcha.

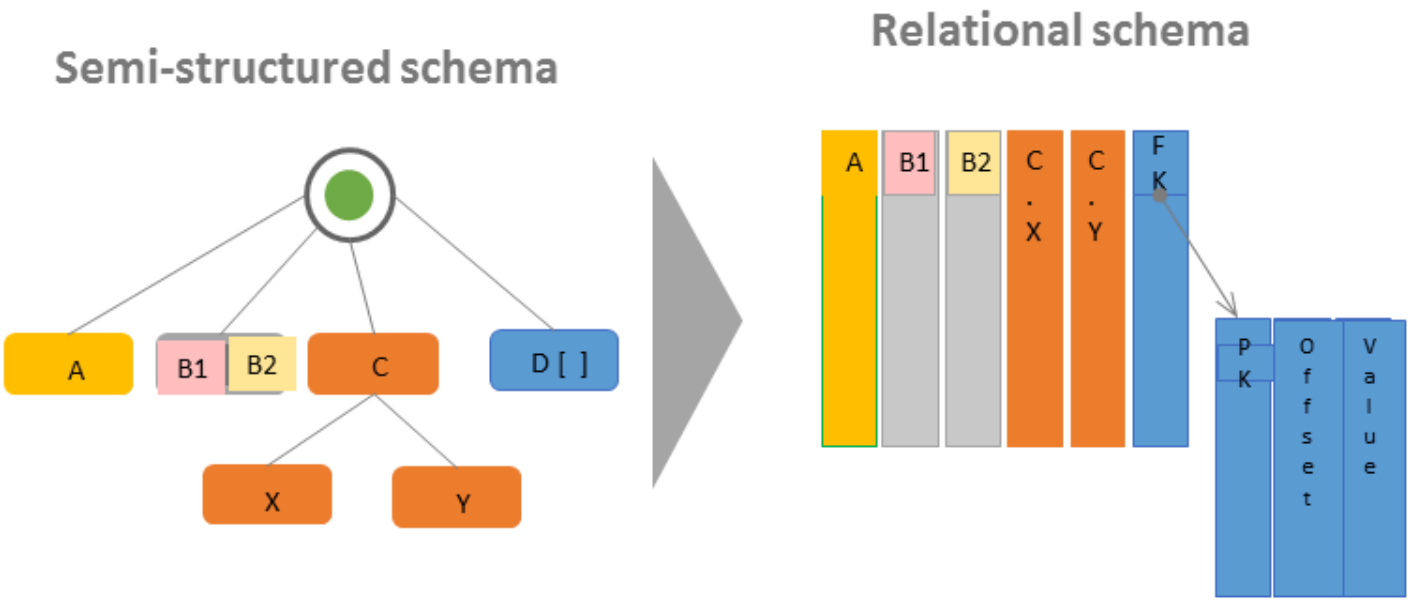
Los datos semiestructurados suelen contar con margen para identificar las entidades dentro de los datos. Puede tener estructuras de datos anidadas sin esquema fijo. Para obtener más información acerca de los datos semiestructurados, consulte Datos semiestructurados en Wikipedia.

Los datos relacionales vienen representados por tablas que constan de filas y columnas. Las relaciones entre las tablas se pueden representar mediante una relación de clave principal a clave externa. Para obtener más información, consulte Base de datos relacional en Wikipedia.

AWS Glue usa los rastreadores para inferir esquemas para los datos semiestructurados. A continuación, transforma los datos en un esquema relacional mediante un flujo de trabajo de ETL

(extracción, transformación y carga). Por ejemplo, es posible que desee diseccionar datos JSON desde archivos de origen de Amazon Simple Storage Service (Amazon S3) hacia tablas de Amazon Relational Database Service (Amazon RDS). Saber cómo controla AWS Glue las diferencias entre los esquemas puede ayudarle a entender el proceso de transformación.

En este diagrama se muestra cómo transforma AWS Glue un esquema semiestructurado en un esquema relacional.



En el siguiente diagrama se ilustra lo siguiente:

- El valor único A se convierte directamente en una columna relacional.
- El par de valores B1 y B2 se convierten en dos columnas relacionales.
- La estructura C, con los elementos secundarios X e Y, se convierte en dos columnas relacionales.

- La matriz `D[]` se convierte en una columna relacional con una clave externa que apunta a otra tabla relacional. Junto con una clave principal, la segunda tabla relacional tiene columnas que contienen el desplazamiento y el valor de los elementos en la matriz.

AWS Sistemas tipo Glue

AWS Glue utiliza varios sistemas de tipos para proporcionar una interfaz versátil en los sistemas de datos que almacenan datos de formas muy diferentes. Este documento elimina la ambigüedad de los sistemas tipo AWS Glue y los estándares de datos.

AWS Tipos de catálogos de datos de Glue

El catálogo de datos es un registro de tablas y campos almacenados en diferentes sistemas de datos, un metaalmacén. Cuando los componentes de AWS AWS Glue, como los rastreadores de AWS Glue y los trabajos de Glue with Spark, escriben en el catálogo de datos, lo hacen con un sistema de tipos interno para rastrear los tipos de campos. Estos valores se muestran en la columna Tipo de datos del esquema de tabla de la AWS Glue Console. Este sistema de tipos se basa en el sistema de tipos de Apache Hive. Para obtener más información sobre el sistema de tipos de Apache Hive, consulte Tipos en la wiki de Apache Hive. Para obtener más información sobre tipos específicos y compatibilidad, se proporcionan ejemplos en la consola AWS Glue, como parte del Generador de esquemas.

Validación, compatibilidad y otros usos

El catálogo de datos no valida los tipos escritos en campos de tipos. Cuando los componentes de AWS Glue lean y escriban en el catálogo de datos, serán compatibles entre sí. AWS Los componentes de Glue también tienen como objetivo preservar un alto grado de compatibilidad con los tipos Hive. Sin embargo, los componentes de AWS Glue no garantizan la compatibilidad con todos los tipos de colmenas. Esto permite la interoperabilidad con herramientas como Athena DDL cuando se trabaja con tablas del catálogo de datos.

Dado que el catálogo de datos no valida los tipos, otros servicios pueden usarlo para rastrear los tipos mediante sistemas que se ajusten estrictamente al sistema de tipos de Hive o a cualquier otro sistema.

Escribe guiones en AWS Glue with Spark

Cuando un script de AWS Glue with Spark interpreta o transforma un conjunto de datos`DynamicFrame`, proporcionamos una representación en memoria del conjunto de datos tal

como se usa en el script. El objetivo de un `DynamicFrame` es similar al del Spark `DataFrame`: modela tu conjunto de datos para que Spark pueda programar y ejecutar transformaciones en los datos. Garantizamos que la representación de tipos de `DynamicFrame` es intercompatible con `DataFrame` y proporciona los métodos `toDF` y `fromDF`.

Si la información de tipos se puede inferir o proporcionar a un `DataFrame`, se puede inferir o proporcionar a un `DynamicFrame`, a menos que se documente lo contrario. Cuando proporcionamos lectores o escritores optimizados para formatos de datos específicos, si Spark puede leer o escribir sus datos, los lectores y escritores que proporcionemos podrán hacerlo, con sujeción a las limitaciones documentadas. Para obtener más información acerca de los lectores y los escritores, consulte the section called “Opciones de formato de datos”.

El tipo Choice

Los `DynamicFrames` proporcionan un mecanismo para modelar los campos de un conjunto de datos cuyo valor puede tener tipos inconsistentes en el disco en todas las filas. Por ejemplo, un campo puede contener un número almacenado como cadena en determinadas filas y un entero en otras. Este mecanismo es de un tipo en memoria denominado `Choice`. Proporcionamos transformaciones, como el `ResolveChoice` método, para convertir las columnas de Choice en un tipo concreto. AWS Glue ETL no escribirá el tipo Choice en el catálogo de datos en el transcurso normal de la operación; los tipos Choice solo existen en el contexto de los modelos de DynamicFrame memoria de los conjuntos de datos. Para ver un ejemplo del uso del tipo Choice, consulte the section called “Muestra de preparación de datos”.

AWS Tipos de Glue Crawler

Los rastreadores tienen como objetivo producir un esquema coherente y utilizable para su conjunto de datos y, después, almacenarlo en el catálogo de datos para usarlo en otros componentes de AWS Glue y Athena. Los rastreadores se encargan de los tipos, como se describe en la sección anterior sobre el catálogo de datos, the section called “AWS Tipos de catálogos de datos de Glue”. Para generar un tipo utilizable en los escenarios de tipo “Choice”, en los que una columna contiene valores de dos o más tipos, los rastreadores crearán un tipo `struct` que modele los tipos potenciales.

Introducción a AWS Glue

En las siguientes secciones, se ofrece información acerca de la configuración de AWS Glue. No todas las secciones de configuración son necesarias para empezar a utilizar AWS Glue. Puede utilizar las instrucciones según sea necesario para configurar los permisos de IAM, el cifrado y el DNS (si utiliza una VPC, un entorno para acceder a los almacenes de datos o si utiliza sesiones interactivas).

Temas

- Información general para el uso de AWS Glue
- Configuración de permisos de IAM para AWS Glue
- Configuración de perfiles AWS Glue de uso
- Introducción al AWS Glue Data Catalog
- Configuración del acceso de red a los almacenes de datos
- Configuración del cifrado en AWS Glue
- Configuración de redes para el desarrollo de AWS Glue

Información general para el uso de AWS Glue

Con AWS Glue se almacenan metadatos en el AWS Glue Data Catalog. Estos metadatos se utilizan para organizar los trabajos de ETL que transforman los orígenes de datos y cargan su almacenamiento de datos o lago de datos. A continuación se describe el flujo de flujo de trabajo general y algunas de las elecciones que se realizan cuando se trabaja con AWS Glue.

> **Note**
>
> Puede seguir los pasos que se indican a continuación o puede crear un flujo de trabajo que realice automáticamente los pasos 1 a 3. Para obtener más información, consulte the section called “Realización de actividades de ETL complejas mediante esquemas y flujos de trabajo”.

1. Rellene AWS Glue Data Catalog con definiciones de tabla.

   En la consola, para almacenes de datos persistentes, puede agregar un rastreador para rellenar AWS Glue Data Catalog. Puede iniciar el asistente para Add crawler (Añadir rastreador) desde la

lista de tablas o la lista de rastreadores. Debe elegir uno o varios almacenes de datos a los que el rastreador tendrá acceso. También puede crear un programa para establecer la frecuencia de ejecución de su rastreador. Para transmisiones de datos, puede crear manualmente la definición de tabla y definir las propiedades de transmisión.

Opcionalmente, puede proporcionar un clasificador personalizado que deduce el esquema de los datos. Puede crear clasificadores personalizados utilizando un patrón de grok de . Por otra parte, AWS Glue proporciona clasificadores integrados que los rastreadores utilizan automáticamente si un clasificador personalizado no reconoce los datos. Cuando define un rastreador, no tiene que seleccionar un clasificador. Para obtener más información sobre los clasificadores de AWS Glue, consulte Adición de clasificadores a un rastreador en AWS Glue.

Para rastrear algunos tipos de almacenes de datos se necesita una conexión que proporcione información de autenticación y de ubicación. Si es necesario, puede crear una conexión que proporcione esta información en la consola de AWS Glue.

El rastreador lee el almacén de datos y crea definiciones de datos y tablas con nombre en AWS Glue Data Catalog. Estas tablas se organizan en una base de datos de su elección. También puede rellenar el Data Catalog con tablas creadas en forma manual. Con este método proporciona el esquema y otros metadatos para crear definiciones de tabla en Data Catalog. Dado que este método puede ser un poco tedioso y dar lugar a errores, a menudo es mejor que un rastreador cree las definiciones de tabla.

Para obtener más información acerca de cómo rellenar AWS Glue Data Catalog con definiciones de tabla, consulte Creación de tablas.

2. Defina un flujo de trabajo que describa la transformación de los datos de origen a destino.

   Por lo general, para crear un flujo de trabajo, tiene que realizar las tareas siguientes:

   - Elegir una tabla de AWS Glue Data Catalog para que sea el origen del trabajo. Su flujo de trabajo utilizará esta definición de tabla para obtener acceso a su origen de datos e interpretar el formato de los datos.
   - Elegir una tabla o ubicación de AWS Glue Data Catalog para que sea el destino del flujo de trabajo. Su flujo de trabajo utilizará esta información para obtener acceso a su almacén de datos.
   - Indicar a AWS Glue para que genere un script para transformar el origen en destino. AWS Glue genera el código para llamar a las transformaciones integradas para que conviertan datos de su esquema de origen en formato de esquema de destino. Estas transformaciones llevan a cabo

operaciones como copiar datos, cambiar el nombre de columnas y filtrar datos para transformar los datos según sea necesario. Puede modificar este script en la consola de AWS Glue.

Para obtener más información acerca de cómo definir trabajos en AWS Glue, consulte Creación de trabajos de ETL visuales con AWS Glue Studio.

3. Ejecute su flujo de trabajo para transformar los datos.

   Puede ejecutar su flujo de trabajo bajo demanda o comenzarlo basándose en uno de los tipos de disparador siguientes:

   - Un disparador basado en un programa cron.
   - Un disparador basado en eventos; por ejemplo, la finalización correcta de otro flujo de trabajo puede iniciar otro flujo de trabajo de AWS Glue.
   - Un disparador que inicie un flujo de trabajo bajo demanda.

   Para obtener más información sobre de los disparadores de AWS Glue, consulte Inicio de trabajos y rastreadores mediante desencadenadores.

4. Monitoree su rastreadores programados y los trabajos activados.

   Use la consola de AWS Glue para ver los elementos siguientes:

   - Detalles y errores de la ejecución del flujo de trabajo.
   - Detalles y errores de la ejecución del rastreador.
   - Todas las notificaciones sobre las actividades de AWS Glue.

   Para obtener más información sobre la monitorización de los rastreadores y los trabajos en AWS Glue, consulte Supervisión de AWS Glue.

Configuración de permisos de IAM para AWS Glue

Las instrucciones de este tema ayudan a configurar rápidamente los permisos AWS Identity and Access Management (IAM) para AWS Glue. Deberá completar las tareas siguientes:

- Conceda a sus identidades de IAM acceso a los recursos AWS Glue.
- Cree un rol de servicio para ejecutar trabajos, acceder a los datos y ejecutar tareas de calidad de datos AWS Glue.

Para obtener instrucciones detalladas que puede utilizar para personalizar los permisos de IAM para AWS Glue, consulte Configuración de permisos de IAM para AWS Glue.

Para configurar los permisos de IAM para AWS Glue en la AWS Management Console

1. Inicie sesión en la AWS Management Console y abra la consola de AWS Glue en https://console.aws.amazon.com/glue/.
2. Elija Empezar.
3. En Preparar su cuenta para AWS Glue, seleccione Configurar permisos de IAM.
4. Elige las identidades de IAM (roles o usuarios) a las que quieres conceder permisos AWS Glue. AWS Glue adjunta la política gestionada de `AWSGlueConsoleFullAccess` a estas identidades. Puede omitir este paso si desea configurar estos permisos de forma manual o solo desea establecer un rol de servicio predeterminado.
5. Elija Siguiente.
6. Elija el nivel de acceso a Amazon S3 que necesitan sus roles y usuarios. Las opciones que elija en este paso se aplican a todas las identidades que haya seleccionado.

   a. En Elegir ubicaciones de S3, elija las ubicaciones de Amazon S3 a las que quiere conceder acceso.

   b. A continuación, seleccione si sus identidades deben tener acceso de Solo lectura (recomendado) o acceso de lectura y escritura a las ubicaciones que seleccionó anteriormente. AWS Glue agrega políticas de permisos a sus identidades en función de la combinación de ubicaciones y los permisos de lectura o escritura que seleccione.

   En la siguiente tabla se muestran los permisos que AWS Glue adjunta al acceso a Amazon S3.

Si elige...	AWS Glue adjunta...
Sin cambios	Sin permisos. AWS Glue no realizará ningún cambio en los permisos de su identidad.
Conceder acceso a ubicaciones específicas de Amazon S3 (solo lectura)	Una política en línea integrada en las identidades de IAM seleccionadas. Para obtener más información, consulte

Si elige...	AWS Glue adjunta...
	Creación de políticas en línea en la Guía del usuario de IAM. AWS Glue nombra la política mediante la siguiente convención:. `AWSGlueConsole` *`<Role/User>`* `InlinePolicy-read-specific-access-` *`<UUID>`* Por ejemplo: `AWSGlueConsoleRole InlinePolicy-read-specific-access-123456780123` . El siguiente es un ejemplo de una política en línea que AWS Glue adjunta para conceder acceso de solo lectura a una ubicación específica de Amazon S3. `{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": [ "s3:Get*", "s3:List*" ], "Resource": [ "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*" ] } ] }`

Si elige...	AWS Glue adjunta...
Conceder acceso a ubicaciones específicas de Amazon S3 (lectura y escritura)	Una política en línea integrada en las identidades de IAM seleccionadas. Para obtener más información, consulte Creación de políticas en línea en la Guía del usuario de IAM. AWS Glue nombra la política mediante la siguiente convención:. `AWSGlueConsole <Role/User> InlinePolicy-read -and-write-specific-access- <UUID>` Por ejemplo: `AWSGlueConsoleRoleInlinePolicy-read-and-write-specific-access-123456780123` . El siguiente es un ejemplo de una política en línea que AWS Glue adjunta para conceder acceso de lectura y escritura a ubicaciones específicas de Amazon S3. `{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": [ "s3:Get*", "s3:List*", "s3:*Object*" ], "Resource": [ "arn:aws:s3:::DOC-EXAMPLE-BUCKET1/*", "arn:aws:s3:::DOC-EXAMPLE-BUCKET2/*" ] } ]`

Si elige...	AWS Glue adjunta...
	`}`
Otorgue el pleno acceso a Amazon S3 (solo lectura)	Política de IAM administrada de `AmazonS3ReadOnlyAccess`. Para obtener más información, consulte la política administrada de AWS: AmazonS3ReadOnlyAccess.
Otorgue acceso completo a Amazon S3 (lectura y escritura)	Política de IAM administrada de `AmazonS3FullAccess`. Para obtener más información, consulte la política administrada de AWS: AmazonS3FullAccess.

7. Elija Siguiente.
8. Elija un rol de servicio predeterminado de AWS Glue para su cuenta. Un rol de servicio es un rol de IAM que AWS Glue utiliza para acceder a los recursos de otros servicios de AWS en su nombre. Para obtener más información, consulte Roles de servicio para AWS Glue.

   - Al elegir la función de rol de servicio de AWS Glue, AWS Glue crea un nuevo rol de IAM en su Cuenta de AWS llamado `AWSGlueServiceRole` con las siguientes políticas gestionadas adjuntas. Si su cuenta ya tiene un rol de IAM denominado `AWSGlueServiceRole`, AWS Glue adjunta estas políticas al rol existente.
     - AWSGlueServiceRole
     - AmazonS3FullAccess
   - Al elegir un rol de IAM existente, AWS Glue lo establece como predeterminado, pero no agrega ningún permiso. Asegúrese de haber configurado el rol para usarlo como rol de servicio para AWS Glue. Para obtener más información, consulte Paso 1: Crear una política de IAM para el servicio AWS Glue y Paso 2: creación de un rol de IAM para AWS Glue.

9. Elija Siguiente.
10. Por último, revise los permisos que ha seleccionado y, a continuación, seleccione Aplicar cambios. Al aplicar los cambios, AWS Glue agregue permisos de IAM a las identidades que

seleccionó. Puede ver o modificar los nuevos permisos en la consola de IAM en https://console.aws.amazon.com/iam/.

Ya ha completado la configuración mínima de permisos de IAM para AWS Glue. En un entorno de producción, recomendamos que se familiarice con la Seguridad en AWS Glue y Gestión de identidad y acceso para AWS Glue para ayudarlo a proteger los recursos de AWS para su caso de uso.

Siguientes pasos

Ahora que ha configurado los permisos de IAM, puede explorar los siguientes temas para empezar a utilizar AWS Glue:

- Primeros pasos con AWS Glue Skill Builder de AWS
- Introducción al AWS Glue Data Catalog

Configuración de AWS Glue Studio

Complete las tareas de esta sección cuando utilice AWS Glue para operaciones de ETL visuales por primera vez:

Temas

- Revisar los permisos de IAM necesarios para el usuario de AWS Glue Studio
- Revisar los permisos de IAM necesarios para trabajos de ETL.
- Definición de permisos de IAM para AWS Glue Studio
- Configurar una VPC para su trabajo de ETL

Revisar los permisos de IAM necesarios para el usuario de AWS Glue Studio

Para utilizar AWS Glue Studio, el usuario debe tener acceso a diversos recursos de AWS. El usuario debe poder ver y seleccionar buckets de Amazon S3, políticas y roles de IAM, y objetos de AWS Glue Data Catalog.

Permisos de servicios de AWS Glue

AWS Glue Studio utiliza las acciones y recursos del servicio de AWS Glue. Su usuario necesita permisos sobre estas acciones y recursos para utilizar AWS Glue Studio de manera eficaz. Puede

conceder al usuario de AWS Glue Studio la política administrada de `AWSGlueConsoleFullAccess` o crear una política personalizada con un conjunto de permisos más pequeño.

> **Important**
>
> Según las mejores prácticas de seguridad, se recomienda restringir el acceso mediante políticas más estrictas para limitar aún más el acceso al bucket de Amazon S3 y grupos de registros de Amazon CloudWatch. Para ver un ejemplo de política de Amazon S3, consulte Cómo escribir políticas de IAM: cómo conceder acceso a un bucket de Amazon S3.

Creación de políticas de IAM personalizadas para AWS Glue Studio

Puede crear una política personalizada con un conjunto de permisos más pequeño para AWS Glue Studio. La política puede conceder permisos para un subconjunto de objetos o acciones. Utilice la siguiente información al crear una política personalizada.

Para utilizar las API de AWS Glue Studio, incluye `glue:UseGlueStudio` en la política de acción de los permisos de IAM. Utilizar `glue:UseGlueStudio` le permitirá acceder a todas las acciones de AWS Glue Studio, incluso a las que se vayan agregando a la API a lo largo del tiempo.

Acciones de gráficos acíclicos dirigidos (DAG)

- CreateDag
- UpdateDag
- GetDag
- DeleteDag

Acciones de trabajo

- SaveJob
- GetJob
- CreateJob
- DeleteJob
- GetJobs
- UpdateJob

Acciones de ejecución de trabajo

- StartJobRun
- GetJobRuns
- BatchStopJobRun
- GetJobRun
- QueryJobRuns
- QueryJobs
- QueryJobRunsAggregated

Acciones de esquemas

- GetSchema
- GetInferredSchema

Acciones de la base de datos

- GetDatabases

Acciones del plan

- GetPlan

Acciones de la tabla

- SearchTables
- GetTables
- GetTable

Acciones de conexión

- CreateConnection
- DeleteConnection
- UpdateConnection

- GetConnections
- GetConnection

Acciones de asignación

- GetMapping

Acciones de proxy de S3

- ListBuckets
- ListObjectsV2
- GetBucketLocation

Acciones de configuración de seguridad

- GetSecurityConfigurations

Acción de script

- CreateScript (diferente de la API del mismo nombre en AWS Glue)

Acceso a API de AWS Glue Studio

Para acceder a AWS Glue Studio, agregue `glue:UseGlueStudio` a la lista de políticas de acciones de los permisos de IAM.

En el siguiente ejemplo, se ha incluido `glue:UseGlueStudio` en la política de acción, pero las API de AWS Glue Studio no están identificadas individualmente. Esto se debe a que cuando se incluye `glue:UseGlueStudio` se concede automáticamente acceso a las API internas sin tener que especificar las API de AWS Glue Studio individuales en los permisos de IAM.

En el ejemplo, las políticas de acción adicionales que aparecen (por ejemplo, `glue:SearchTables`) no son API de AWS Glue Studio, de modo que se deberán incluir en los permisos de IAM según sea necesario. También se pueden incluir acciones de proxy de Amazon S3 para especificar el nivel de acceso de Amazon S3 que se desea conceder. La siguiente política de ejemplo proporciona acceso para abrir AWS Glue Studio, crear un trabajo visual y guardarlo o ejecutarlo si el rol de IAM seleccionado tiene suficiente acceso.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "glue:UseGlueStudio",
                "iam:ListRoles",
                "iam:ListUsers",
                "iam:ListGroups",
                "iam:ListRolePolicies",
                "iam:GetRole",
                "iam:GetRolePolicy",
                "glue:SearchTables",
                "glue:GetConnections",
                "glue:GetJobs",
                "glue:GetTables",
                "glue:BatchStopJobRun",
                "glue:GetSecurityConfigurations",
                "glue:DeleteJob",
                "glue:GetDatabases",
                "glue:CreateConnection",
                "glue:GetSchema",
                "glue:GetTable",
                "glue:GetMapping",
                "glue:CreateJob",
                "glue:DeleteConnection",
                "glue:CreateScript",
                "glue:UpdateConnection",
                "glue:GetConnection",
                "glue:StartJobRun",
                "glue:GetJobRun",
                "glue:UpdateJob",
                "glue:GetPlan",
                "glue:GetJobRuns",
                "glue:GetTags",
                "glue:GetJob",
                "glue:QueryJobRuns",
                "glue:QueryJobs",
                "glue:QueryJobRunsAggregated"
            ],
            "Resource": "*"
```

```
        },
        {
            "Action": [
                "iam:PassRole"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:iam::*:role/AWSGlueServiceRole*",
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": [
                        "glue.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

Permisos de vista previa de datos y cuaderno

Las vistas previas de datos y los cuaderno le permiten ver una muestra de los datos en cualquier etapa del trabajo (lectura, transformación, escritura), sin tener que ejecutar el trabajo. Se especifica un rol de AWS Identity and Access Management (IAM) para que utilice AWS Glue Studio al acceder a los datos. Los roles de IAM están destinados a ser asumibles y no tienen asociadas credenciales estándar a largo plazo, como una contraseña o clave de acceso. En lugar de esto, cuando AWS Glue Studio asume el rol, IAM le proporciona credenciales de seguridad temporales.

Para garantizar que las vistas previas de datos y los comandos de cuaderno funcionen correctamente, utilice un rol que tenga un nombre que empiece por la cadena `AWSGlueServiceRole`. Si decide utilizar otro nombre para el rol, debe agregar el permiso `iam:passrole` y configurar una política para el rol en IAM. Para obtener más información, consulte Crear una política de IAM para roles no denominados “AWSGlueServiceRole*”.

> **Warning**
>
> Si un rol concede el permiso `iam:passrole` a un cuaderno e implementa el encadenamiento de roles, un usuario podría obtener acceso involuntariamente al cuaderno.

Actualmente no se ha implementado ninguna auditoría que le permita monitorear a qué usuarios se les ha concedido acceso al cuaderno.

Si desea denegar a una identidad de IAM la capacidad de crear sesiones de vista previa de datos, consulte el siguiente ejemplo the section called “Cómo denegar a una identidad la capacidad de crear sesiones de vista previa de datos”.

Permisos de Amazon CloudWatch

Puede monitorear sus trabajos de AWS Glue Studio mediante Amazon CloudWatch, que recopila y procesa los datos sin procesar de AWS Glue en métricas legibles y casi en tiempo real. De forma predeterminada, los datos de las métricas de AWS Glue se envían a CloudWatch en forma automática. Para obtener más información, consulte ¿Qué es Amazon CloudWatch? en la Guía del usuario de Amazon CloudWatch, y Métricas de AWS Glue en la Guía para desarrolladores de AWS Glue.

Para acceder a los paneles de CloudWatch, el usuario que accede a AWS Glue Studio necesita alguna de las siguientes:

- La política `AdministratorAccess`
- La política `CloudWatchFullAccess`
- Una política personalizada que incluya uno o varios de estos permisos específicos:
  - `cloudwatch:GetDashboard` y `cloudwatch:ListDashboards` para ver paneles
  - `cloudwatch:PutDashboard` para poder crear o modificar paneles
  - `cloudwatch:DeleteDashboards` para eliminar paneles

Para obtener más información sobre cómo cambiar los permisos de un usuario de IAM mediante políticas, consulte Cambio de los permisos de un usuario de IAM en la Guía del usuario de IAM.

Revisar los permisos de IAM necesarios para trabajos de ETL.

Cuando crea un trabajo con AWS Glue Studio, el trabajo asume los permisos del rol de IAM que se especifica al crear el trabajo. Este rol de IAM debe tener permiso para extraer datos de su origen de datos, escribir en su destino y acceder a recursos de AWS Glue.

El nombre del rol que cree para el trabajo debe comenzar con la cadena `AWSGlueServiceRole` para que AWS Glue Studio lo utilice correctamente. Por ejemplo, podría asignar el siguiente nombre a su rol: `AWSGlueServiceRole-FlightDataJob`.

Permisos de origen de datos y destino de datos

Un trabajo de AWS Glue Studio debe tener acceso a Amazon S3 para todos los orígenes, destinos, scripts y directorios temporales que utilice en su trabajo. Puede crear una política para proporcionar un acceso detallado a determinados recursos de Amazon S3.

- Los orígenes de datos requieren permisos `s3:ListBucket` y `s3:GetObject`.
- Los destinos de datos requieren permisos `s3:ListBucket`, `s3:PutObject` y `s3:DeleteObject`.

Si elige Amazon Redshift como origen de datos, puede proporcionar un rol para los permisos de clúster. Los trabajos que se ejecutan respecto de un clúster de Amazon Redshift envían comandos que acceden a Amazon S3 para el almacenamiento temporal mediante credenciales temporales. Si el trabajo se ejecuta durante más de una hora, estas credenciales caducarán, y provocarán un error en el trabajo. Para evitar este problema, puede asignar un rol al clúster de Amazon Redshift que conceda los permisos necesarios a los trabajos que utilizan credenciales temporales. Para obtener más información, consulte Movimiento de datos desde y hacia Amazon Redshift en la Guía para desarrolladores de AWS Glue.

Si el trabajo utiliza orígenes o destinos de datos distintos de Amazon S3, debe asociar los permisos necesarios al rol de IAM utilizado por el trabajo para acceder a estos orígenes y destinos de datos. Para obtener más información, consulte Configuración del entorno para obtener acceso a almacenes de datos en la Guía para desarrolladores de AWS Glue.

Si utiliza conectores y conexiones para el almacén de datos, necesita permisos adicionales, como se describe en the section called “Permisos necesarios para utilizar conectores”.

Permisos necesarios para eliminar trabajos

En AWS Glue Studio, puede seleccionar múltiples trabajos para eliminar en la consola. Para realizar esta acción, debe tener el permiso `glue:BatchDeleteJob`. Esto es diferente de la consola de AWS Glue, que requiere el permiso `glue:DeleteJob` para eliminar trabajos.

Permisos de AWS Key Management Service

Si tiene previsto acceder a orígenes y destinos de Amazon S3 que utilizan cifrado del lado del servidor con AWS Key Management Service (AWS KMS), adjunte una política al rol de AWS Glue Studio utilizado por el trabajo que permita al trabajo descifrar los datos. El rol del trabajo necesita los permisos `kms:ReEncrypt`, `kms:GenerateDataKey` y `kms:DescribeKey`. Además, el rol del trabajo necesita el permiso `kms:Decrypt` para cargar o descargar un objeto de Amazon S3 cifrado con una clave maestra del cliente (CMK) AWS KMS.

La utilización de CMK de AWS KMS conlleva cargos adicionales. Para obtener más información, consulte Conceptos de AWS Key Management Service: claves maestras del cliente (CMK) y Precios de AWS Key Management Service en la Guía para desarrolladores de AWS Key Management Service.

Permisos necesarios para utilizar conectores

Si utiliza un conector personalizado y conexión de AWS Glue para acceder a un almacén de datos, el rol utilizado para ejecutar el trabajo de ETL de AWS Glue necesita permisos adicionales asociados:

- La política administrada de AWS `AmazonEC2ContainerRegistryReadOnly` para acceder a los conectores adquiridos en AWS Marketplace.
- Los permisos `glue:GetJob` y `glue:GetJobs`.
- Los permisos de AWS Secrets Manager para acceder a los secretos que se utilizan con las conexiones. Consulte Ejemplo: Permiso para recuperar valores de secretos para ver ejemplos de políticas de IAM.

Si sus trabajos de ETL de AWS Glue se ejecutan en una VPC que ejecuta Amazon VPC, la VPC debe configurarse como se describe en the section called “Configurar una VPC para su trabajo de ETL”.

Definición de permisos de IAM para AWS Glue Studio

Puede crear los roles y asignar políticas a usuarios y roles de trabajo mediante el usuario administrado de AWS.

Puede utilizar la política administrada de AWS AWSGlueConsoleFullAccess para proporcionar los permisos necesarios para utilizar la consola de AWS Glue Studio.

Para crear su propia política, siga los pasos documentados en Crear una política de IAM para el servicio AWS Glue en la Guía para desarrolladores de AWS Glue. Incluya los permisos de IAM

descritos anteriormente en Revisar los permisos de IAM necesarios para el usuario de AWS Glue Studio.

Temas

- Adjuntar políticas al usuario de AWS Glue Studio
- Crear una política de IAM para roles no denominados “AWSGlueServiceRole*”

Adjuntar políticas al usuario de AWS Glue Studio

Todo usuario de AWS que inicie sesión en la consola de AWS Glue Studio debe tener permisos para acceder a recursos específicos. Estos permisos los concede mediante la asignación de políticas de IAM al usuario.

Para adjuntar la política administrada AWSGlueConsoleFullAccess a un usuario

1. Inicie sesión en la AWS Management Console y abra la consola de IAM en https://console.aws.amazon.com/iam/.
2. En el panel de navegación, seleccione Policies (Políticas).
3. En la lista de políticas, seleccione la casilla de verificación situada junto a AWSGlueConsoleFullAccess. Puede utilizar el menú Filter y el cuadro de búsqueda para filtrar la lista de políticas.
4. Seleccione Policy actions (Acciones de la política) y, a continuación, Attach (Adjuntar).
5. Seleccione el usuario al que asociará la política. Puede utilizar el menú Filter (Filtro) y el cuadro de búsqueda para filtrar la lista entidades principales. Después de seleccionar el usuario al que asociará la política, seleccione Attach policy (Asociar política).
6. Repita los pasos anteriores para adjuntar políticas adicionales al usuario según sea necesario.

Crear una política de IAM para roles no denominados “AWSGlueServiceRole*”

Para configurar una política de IAM para los roles utilizados por AWS Glue Studio

1. Inicie sesión en la AWS Management Console y abra la consola de IAM en https://console.aws.amazon.com/iam/.
2. Agregue una nueva política de IAM. Puede agregar a una política existente o crear una nueva política insertada de IAM. Para crear una política de IAM:

1. Elija Policies y después, Create Policy. Si aparece el botón Get Started (Empezar), elíjalo y, a continuación, elija Create Policy (Crear política).
2. Junto a Create Your Own Policy, seleccione Select.
3. En Policy Name (Nombre de política), escriba cualquier valor que sea fácil de consultar más tarde. Si lo desea, escriba un texto descriptivo en Description (Descripción).
4. En Policy Document (Documento de política), escriba una instrucción de política con el formato siguiente y, a continuación, elija Create Policy (Crear política):

3. Copie y pegue los siguientes bloques en la política en la matriz “Declaración” y sustituya *my-interactive-session-role-prefix* con el prefijo para que todos los roles comunes se asocien con permisos para AWS Glue.

```
{
    "Action": [
        "iam:PassRole"
    ],
    "Effect": "Allow",
    "Resource": "arn:aws:iam::*:role/my-interactive-session-role-prefix*",
    "Condition": {
        "StringLike": {
            "iam:PassedToService": [
                "glue.amazonaws.com "
            ]
        }
    }
}
```

Este es el ejemplo completo con las matrices Version (Versión) y Statement (Declaración) incluidas en la política

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
        "Action": [
            "iam:PassRole"
        ],
        "Effect": "Allow",
        "Resource": "arn:aws:iam::*:role/my-interactive-session-role-prefix*",
        "Condition": {
```

```
            "StringLike": {
                "iam:PassedToService": [
                    "glue.amazonaws.com "
                ]
            }
        }
    }
  ]
}
```

4. Para habilitar la política para un usuario, elija Users (Usuarios).
5. Elija el usuario de a quien desea asociar la política.

Configurar una VPC para su trabajo de ETL

Puede utilizar Amazon Virtual Private Cloud (Amazon VPC) para definir una red virtual en su propia área aislada lógicamente dentro de la Nube de AWS, conocida como una nube privada virtual (VPC). Puede lanzar recursos de AWS, como, por ejemplo, instancias, en su VPC. Una VPC es prácticamente idéntica a una red tradicional que usted puede operar en su propio centro de datos, con los beneficios que supone utilizar la infraestructura escalable de AWS. Puede configurar la VPC, seleccionar su rango de direcciones IP, crear subredes y configurar tablas de ruteo, gateways de red y ajustes de seguridad. Ahora puede conectar sus instancias de la VPC a Internet. Puede conectar la VPC a su propio centro de datos corporativo, lo que convierte la Nube de AWS en una ampliación del centro de datos. Para proteger los recursos de cada subred, puede utilizar varias capas de seguridad, incluidos grupos de seguridad y listas de control de acceso a la red. Para obtener más información, consulte la Guía del usuario de Amazon VPC.

Puede configurar sus trabajo de ETL de AWS Glue para que se ejecuten dentro de una VPC cuando se utilicen conectores. Debe configurar la VPC para lo siguiente, según sea necesario:

- Acceso a la red pública de almacenes de datos externos a AWS. Todos los almacenes de datos a los que obtiene acceso el trabajo deben estar disponibles a partir de la subred de la VPC.
- Si su trabajo necesita obtener acceso a los recursos de la VPC y a la red pública de Internet, la VPC debe tener una gateway de NAT (traducción de direcciones de red) dentro de la VPC.

  Para obtener más información, consulte Configuración del entorno para obtener acceso a almacenes de datos en la Guía para desarrolladores de AWS Glue.

Introducción a los cuaderno en AWS Glue Studio

Al iniciar un cuaderno a través de AWS Glue Studio, todos los pasos de configuración se hacen por usted para que, después de unos segundos, pueda explorar los datos y comenzar a desarrollar el script de trabajo.

Las siguientes secciones describen cómo crear un rol y otorgar los permisos adecuados para utilizar cuadernos en AWS Glue Studio para trabajos de ETL.

Temas

- Concesión de permisos para el rol de IAM

Concesión de permisos para el rol de IAM

Configurar AWS Glue Studio es un requisito previo para utilizar cuadernos.

Para utilizar cuadernos en AWS Glue, la función requiere lo siguiente:

- Una relación de confianza con AWS Glue para la acción `sts:AssumeRole` y, si desea etiquetar, `sts:TagSession`.
- Una política de IAM que contenga todas las operaciones de la API para cuadernos, AWS Glue y sesiones interactivas.
- Una política de IAM para un rol de pase, ya que el rol debe poder pasarse a sí mismo desde el cuaderno a sesiones interactivas.

Por ejemplo, cuando crea un nuevo rol, puede agregar una política administrada de AWS estándar como `AWSGlueConsoleFullAccessRole` al rol y luego agregar una nueva política para las operaciones del cuaderno y otra para la política IAM PassRole.

Acciones necesarias para una relación de confianza con AWS Glue

Al iniciar una sesión de cuaderno, se debe agregar `sts:AssumeRole` a la relación de confianza del rol que se pase al cuaderno. Si la sesión incluye etiquetas, también se debe pasar la acción `sts:TagSession`. Sin estas acciones, la sesión del cuaderno no puede iniciarse.

Por ejemplo:

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "glue.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Políticas que contienen las operaciones de API para cuadernos

La siguiente política de ejemplo describe los permisos de AWS IAM requeridos para cuadernos. Si va a crear un nuevo rol, cree una política que contenga lo siguiente:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:StartNotebook",
                "glue:TerminateNotebook",
                "glue:GlueNotebookRefreshCredentials",
                "glue:DeregisterDataPreview",
                "glue:GetNotebookInstanceStatus",
                "glue:GlueNotebookAuthorize"
            ],
            "Resource": "*"
        }
    ]
}
```

Puede utilizar las siguientes políticas de IAM para permitir el acceso a recursos específicos:

- AwsGlueSessionUserRestrictedNotebookServiceRole: proporciona acceso completo a todos los recursos de AWS Glue, excepto las sesiones. Permite a los usuarios crear y utilizar solo las sesiones de cuadernos que estén asociadas a esos usuarios. Esta política también incluye otros

permisos que AWS Glue necesita para administrar recursos de AWS Glue en otros servicios de AWS.

- AwsGlueSessionUserRestrictedNotebookPolicy: proporciona permisos que permiten a los usuarios crear y utilizar solo las sesiones de cuadernos que estén asociadas a esos usuarios. Esta política también incluye permisos para que los usuarios puedan pasar expresamente un rol de sesión de AWS Glue restringida.

Política de IAM para pasar un rol

Cuando se crea un cuaderno con un rol, ese rol se pasa a las sesiones interactivas para que se pueda utilizar el mismo rol en ambos lugares. Como tal, el permiso `iam:PassRole` debe formar parte de la política del rol.

Cree una nueva política para el rol mediante el siguiente ejemplo. Reemplace el número de cuenta por el suyo y por el nombre del rol.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::090000000210:role/<role_name>"
        }
    ]
}
```

Configuración de perfiles AWS Glue de uso

Una de las principales ventajas de utilizar una plataforma en la nube es su flexibilidad. Sin embargo, esta facilidad a la hora de crear recursos de cómputo conlleva el riesgo de que los costes de la nube aumenten vertiginosamente si no se gestionan ni tienen barreras. Como resultado, los administradores deben encontrar el equilibrio entre evitar los altos costos de infraestructura y, al mismo tiempo, permitir que los usuarios trabajen sin fricciones innecesarias.

Con los perfiles de AWS Glue uso, los administradores pueden crear diferentes perfiles para las distintas clases de usuarios de la cuenta, como desarrolladores, evaluadores y equipos de

productos. Cada perfil es un conjunto único de parámetros que se pueden asignar a distintos tipos de usuarios. Por ejemplo, los desarrolladores pueden necesitar más trabajadores y tener un número máximo de trabajadores más alto, mientras que los equipos de productos pueden necesitar menos trabajadores y un valor de tiempo de espera o inactividad más bajo.

Ejemplo de comportamiento de trabajos y ejecuciones

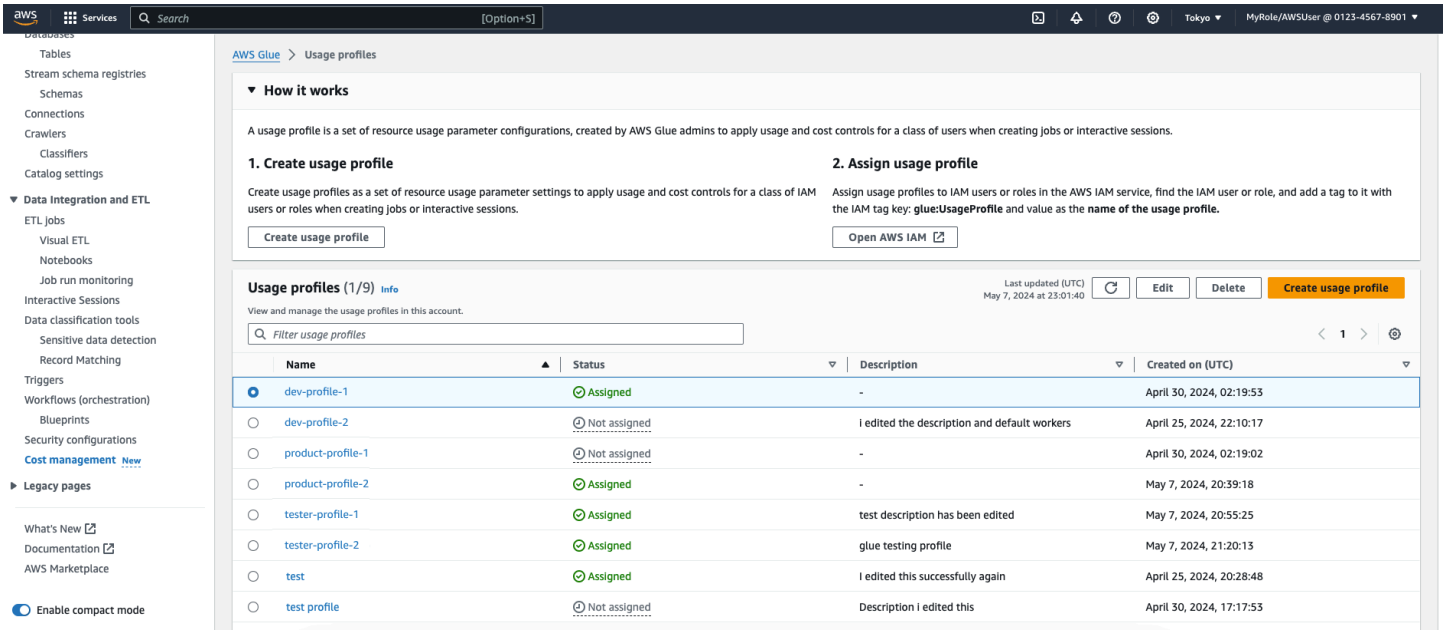
Supongamos que un trabajo lo crea el usuario A con el perfil A. El trabajo se guarda con determinados valores de parámetros. El usuario B con el perfil B intentará ejecutar el trabajo.

Cuando el usuario A creó el trabajo, si no estableció un número específico de trabajadores, se aplicó el conjunto predeterminado en el perfil del usuario A y se guardó con las definiciones del trabajo.

Cuando el usuario B ejecuta el trabajo, lo hace con los valores que se hayan guardado para él. Si el propio perfil del usuario B es más restrictivo y no se le permite ejecutarse con tantos trabajadores, la ejecución de la tarea fallará.

Utilice el perfil como recurso

Un perfil de AWS Glue uso es un recurso identificado por un nombre de recurso de Amazon (ARN). Se aplican todos los controles de IAM (Identity and Access Management) predeterminados, incluida la autorización basada en acciones y en recursos. Los administradores deben actualizar la política de IAM de los usuarios que crean AWS Glue recursos, concediéndoles acceso para usar los perfiles.



Temas

- Creación y administración de perfiles de uso
- Perfiles de uso y trabajos

Creación y administración de perfiles de uso

Creación de un perfil AWS Glue de uso

Los administradores deben crear perfiles de uso y luego asignarlos a los distintos usuarios. Al crear un perfil de uso, debe especificar los valores predeterminados, así como un rango de valores permitidos para varios parámetros de trabajo y sesión. Debe configurar al menos un parámetro para los trabajos o las sesiones interactivas. Puede personalizar el valor predeterminado que se utilizará cuando no se proporcione un valor de parámetro para el trabajo, o configurar un límite de rango o un conjunto de valores permitidos para la validación si un usuario proporciona un valor de parámetro al usar este perfil.

Los valores predeterminados son una práctica recomendada establecida por el administrador para ayudar a los autores de los trabajos. Cuando un usuario crea un nuevo trabajo y no establece un valor de tiempo de espera, se aplicará el tiempo de espera predeterminado del perfil de uso. Si el autor no tiene un perfil, se aplicarán los valores predeterminados del AWS Glue servicio y se guardarán en la definición del trabajo. En tiempo de ejecución, AWS Glue impone los límites establecidos en el perfil (número mínimo, máximo de trabajadores permitidos).

Una vez configurado un parámetro, todos los demás parámetros son opcionales. Los parámetros que se pueden personalizar para los trabajos o las sesiones interactivas son:

- Número de trabajadores: restrinja el número de trabajadores para evitar el uso excesivo de los recursos informáticos. Puede establecer un valor predeterminado, mínimo y máximo. El mínimo es 1.
- Tipo de trabajador: restrinja los tipos de trabajadores relevantes para sus cargas de trabajo. Puede establecer un tipo predeterminado y permitir tipos de trabajadores para un perfil de usuario.
- Tiempo de espera: defina el tiempo máximo durante el que un trabajo o sesión interactiva puede ejecutarse y consumir recursos antes de que finalice. Configure valores de tiempo de espera para evitar trabajos de larga duración.

  Puede establecer un valor predeterminado, mínimo y máximo en minutos. El mínimo es 1 (minuto). Si bien el tiempo de espera AWS Glue predeterminado es de 2880 minutos, puede establecer cualquier valor predeterminado en el perfil de uso.

Se recomienda establecer un valor para «predeterminado». Este valor se utilizará para la creación del trabajo o la sesión si el usuario no ha establecido ningún valor.

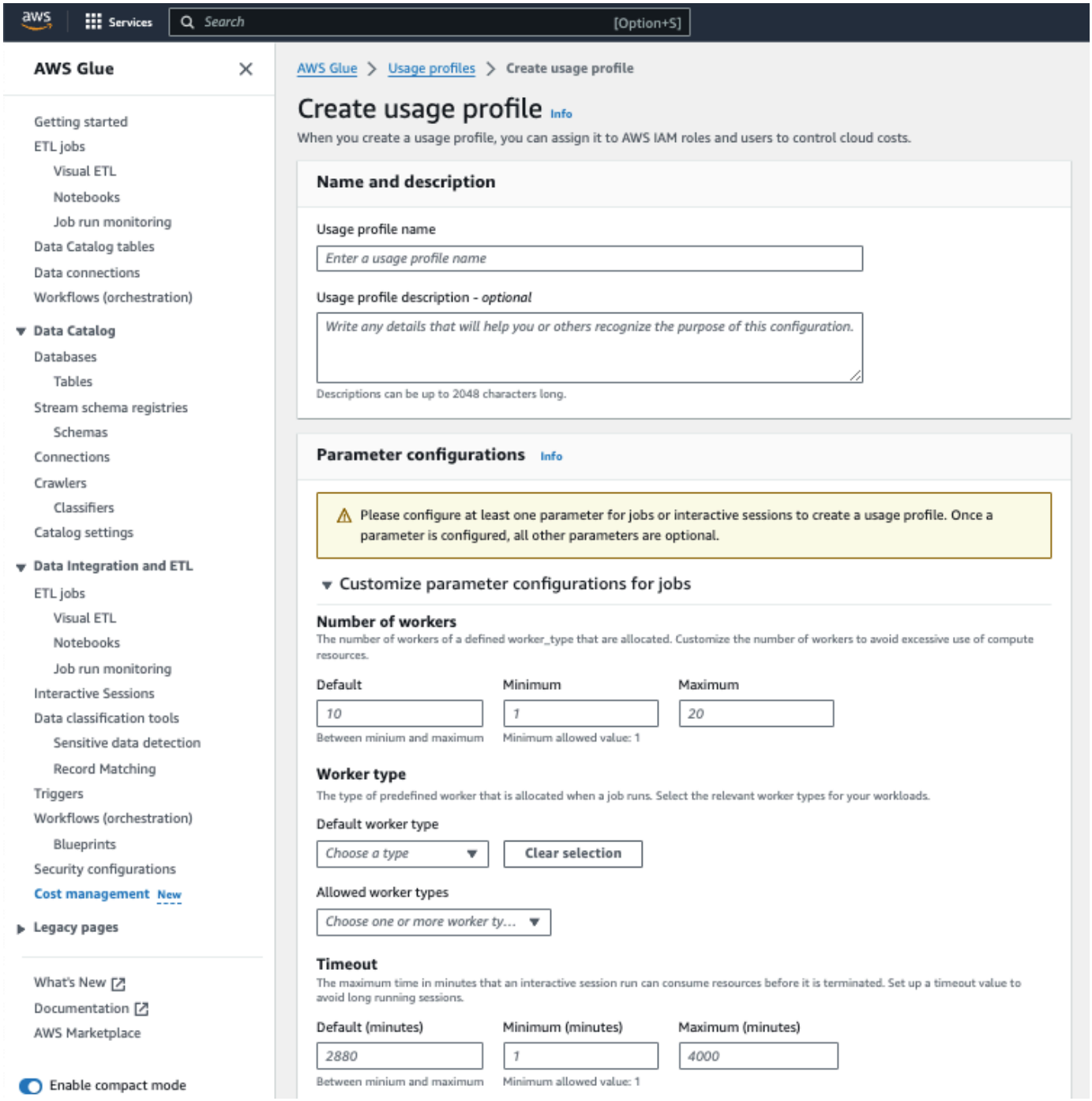
- Tiempo de espera de inactividad: defina el número de minutos que una sesión interactiva permanece inactiva antes de que se agote el tiempo de espera una vez ejecutada una celda. Defina el tiempo de espera de inactividad para que las sesiones interactivas finalicen una vez finalizado el trabajo. El rango de tiempo de espera de inactividad debe estar dentro del límite de tiempo de espera.

  Puede establecer un valor predeterminado, mínimo y máximo en minutos. El mínimo es 1 (minuto). Si bien el tiempo de espera AWS Glue predeterminado es de 2880 minutos, puede establecer cualquier valor predeterminado en el perfil de uso.

  Se recomienda establecer un valor para «predeterminado». Este valor se utilizará para la creación de la sesión si el usuario no ha establecido ningún valor.

Para crear un perfil de AWS Glue uso como administrador (consola)

1. En el menú de navegación de la izquierda, selecciona Gestión de costes.
2. Selecciona Crear perfil de uso.
3. Introduzca el nombre del perfil de uso para el perfil de uso.
4. Introduzca una descripción opcional que ayude a otras personas a reconocer el propósito del perfil de uso.
5. Defina al menos un parámetro en el perfil. Cualquier campo del formulario es un parámetro. Por ejemplo, el tiempo mínimo de espera de inactividad de la sesión.
6. Defina cualquier etiqueta opcional que se aplique al perfil de uso.
7. Seleccione Guardar.



Para crear un perfil de uso (AWS CLI)

1. Escriba el siguiente comando.

```
aws glue create-usage-profile --name profile-name --configuration file://
config.json --tags list-of-tags
```

donde config.json puede definir valores de parámetros para sesiones interactivas (`SessionConfiguration`) y trabajos (`JobConfiguration`):

```
//config.json (There is a separate blob for session/job configuration
{
    "SessionConfiguration": {
        "timeout": {
            "DefaultValue": "2880",
            "MinValue": "100",
            "MaxValue": "4000"
        },
        "idleTimeout": {
            "DefaultValue": "30",
            "MinValue": "10",
            "MaxValue": "4000"
        },
        "workerType": {
            "DefaultValue": "G.2X",
            "AllowedValues": [
                "G.2X",
                "G.4X",
                "G.8X"
            ]
        },
        "numberOfWorkers": {
            "DefaultValue": "10",
            "MinValue": "1",
            "MaxValue": "10"
        }
    },
    "JobConfiguration": {
        "timeout": {
            "DefaultValue": "2880",
            "MinValue": "100",
            "MaxValue": "4000"
        },
        "workerType": {
            "DefaultValue": "G.2X",
            "AllowedValues": [
                "G.2X",
                "G.4X",
                "G.8X"
```

```
            ]
        },
        "numberOfWorkers": {
            "DefaultValue": "10",
            "MinValue": "1",
            "MaxValue": "10"
        }
    }
}
```

2. Ingresa el siguiente comando para ver el perfil de uso creado:

```
aws glue get-usage-profile --name profile-name
```

La respuesta:

```
{
    "ProfileName": "foo",
    "Configuration": {
        "SessionConfiguration": {
            "numberOfWorkers": {
                "DefaultValue": "10",
                "MinValue": "1",
                "MaxValue": "10"
            },
            "workerType": {
                "DefaultValue": "G.2X",
                "AllowedValues": [
                    "G.2X",
                    "G.4X",
                    "G.8X"
                ]
            },
            "timeout": {
                "DefaultValue": "2880",
                "MinValue": "100",
                "MaxValue": "4000"
            },
            "idleTimeout": {
                "DefaultValue": "30",
                "MinValue": "10",
                "MaxValue": "4000"
            }
```

```
        },
        "JobConfiguration": {
            "numberOfWorkers": {
                "DefaultValue": "10",
                "MinValue": "1",
                "MaxValue": "10"
            },
            "workerType": {
                "DefaultValue": "G.2X",
                "AllowedValues": [
                    "G.2X",
                    "G.4X",
                    "G.8X"
                ]
            },
            "timeout": {
                "DefaultValue": "2880",
                "MinValue": "100",
                "MaxValue": "4000"
            }
        }
    },
    "CreatedOn": "2024-01-19T23:15:24.542000+00:00"
}
```

Comandos CLI adicionales utilizados para administrar los perfiles de uso:

- Pegamento AWS list-usage-profiles
- *aws glue update-usage-profile --name profile-name --archivo de configuración: //config.json*
- aws glue delete-usage-profile *--name nombre de perfil*

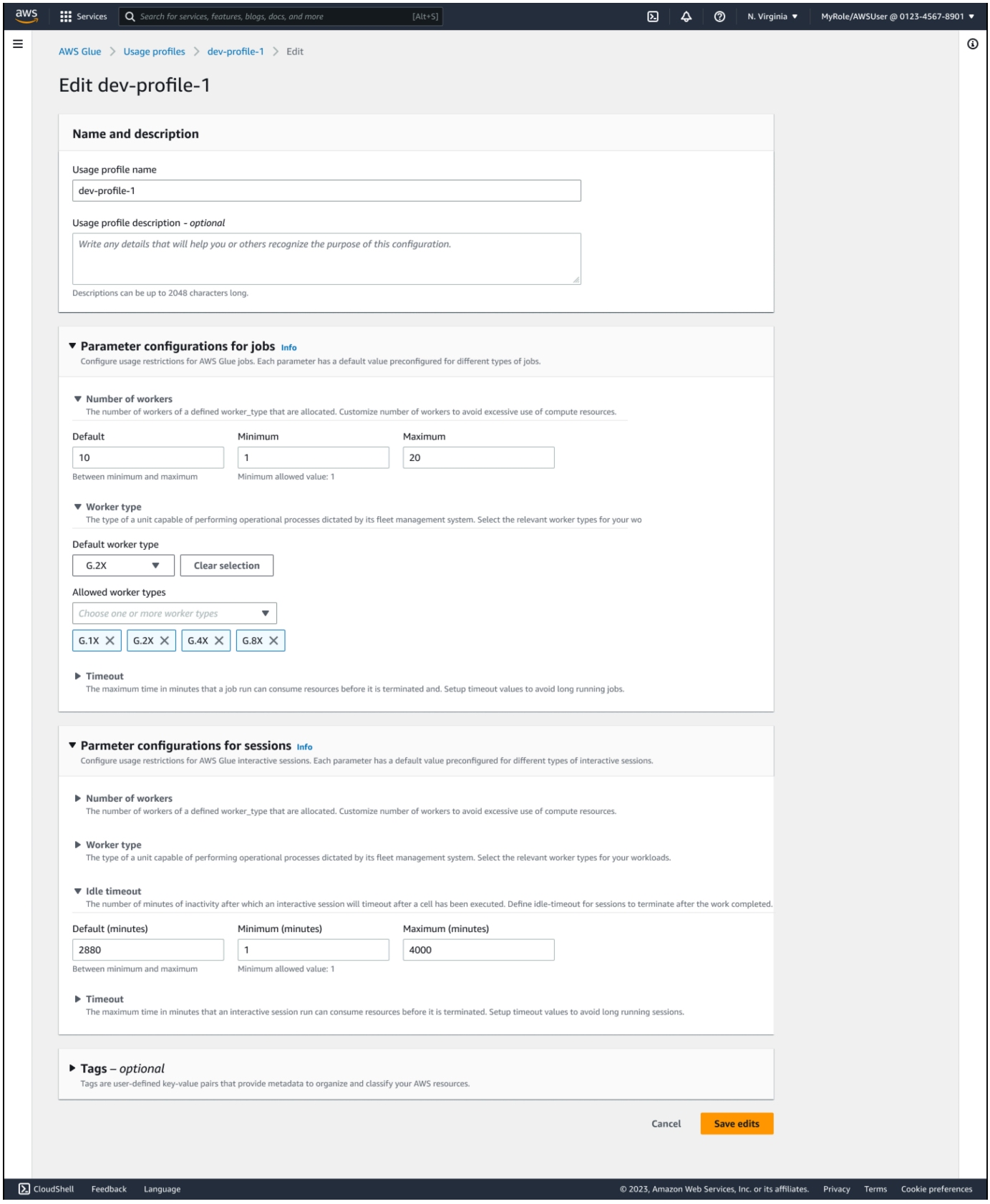
Edición de un perfil de uso

Los administradores pueden editar los perfiles de uso que hayan creado para cambiar los valores de los parámetros del perfil para los trabajos y las sesiones interactivas.

Para editar un perfil de uso:

Para editar un perfil de AWS Glue uso como administrador (consola)

1. En el menú de navegación de la izquierda, selecciona Gestión de costes.
2. Elija un perfil de uso para el que tenga permisos de edición y seleccione Editar.
3. Realice los cambios necesarios en el perfil. Por defecto, los parámetros que ya tienen valores se expanden.
4. Seleccione Guardar ediciones.


Services
Search for services, features, blogs, docs, and more [Alt+S]
N. Virginia
MyRole/AWSUser @ 0123-4567-8901
AWS Glue > Usage profiles > dev-profile-1 > Edit
Edit dev-profile-1
Name and description
Usage profile name
dev-profile-1
Usage profile description - optional
Write any details that will help you or others recognize the purpose of this configuration.
Descriptions can be up to 2048 characters long.
Parameter configurations for jobs Info
Configure usage restrictions for AWS Glue jobs. Each parameter has a default value preconfigured for different types of jobs.
Number of workers
The number of workers of a defined worker_type that are allocated. Customize number of workers to avoid excessive use of compute resources.
Default
10
Between minimum and maximum
Minimum
1
Minimum allowed value: 1
Maximum
20
Worker type
The type of a unit capable of performing operational processes dictated by its fleet management system. Select the relevant worker types for your wo
Default worker type
G.2X
Clear selection
Allowed worker types
Choose one or more worker types
G.1X
G.2X
G.4X
G.8X
Timeout
The maximum time in minutes that a job run can consume resources before it is terminated and. Setup timeout values to avoid long running jobs.
Parmeter configurations for sessions Info
Configure usage restrictions for AWS Glue interactive sessions. Each parameter has a default value preconfigured for different types of interactive sessions.
Number of workers
The number of workers of a defined worker_type that are allocated. Customize number of workers to avoid excessive use of compute resources.
Worker type
The type of a unit capable of performing operational processes dictated by its fleet management system. Select the relevant worker types for your workloads.
Idle timeout
The number of minutes of inactivity after which an interactive session will timeout after a cell has been executed. Define idle-timeout for sessions to terminate after the work completed.
Default (minutes)
2880
Between minimum and maximum
Minimum (minutes)
1
Minimum allowed value: 1
Maximum (minutes)
4000
Timeout
The maximum time in minutes that an interactive session run can consume resources before it is terminated. Setup timeout values to avoid long running sessions.
Tags – optional
Tags are user-defined key-value pairs that provide metadata to organize and classify your AWS resources.
Cancel
Save edits
CloudShell
Feedback
Language
© 2023, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

Para editar un perfil de uso (AWS CLI)

- Escriba el siguiente comando. Se utiliza la misma sintaxis de `--configuration` archivo que se muestra arriba en el comando create.

  ```
  aws glue update-usage-profile --name profile-name --configuration file://
  config.json
  ```

  donde config.json define los valores de los parámetros para las sesiones interactivas (`SessionConfiguration`) y los trabajos (`JobConfiguration`):

Asignación de un perfil de uso

La columna de estado de uso de la página de perfiles de uso muestra si un perfil de uso está asignado a los usuarios. Al pasar el ratón sobre el estado, se muestran las entidades de IAM asignadas.

El administrador puede asignar un perfil de AWS Glue uso a los usuarios o roles que crean recursos. AWS Glue La asignación de un perfil es una combinación de dos acciones:

- Actualizar la etiqueta de usuario o rol de IAM con la clave, luego `glue:UsageProfile`
- Actualización de la política de IAM del usuario o rol.

En el caso de los usuarios que utilizan AWS Glue Studio para crear trabajos o sesiones interactivas, el administrador etiqueta las siguientes funciones:

- Para restringir los trabajos, el administrador etiqueta el rol de consola en el que se ha iniciado sesión
- Para restringir las sesiones interactivas, el administrador etiqueta el rol que el usuario proporciona al crear el bloc de notas

A continuación se muestra un ejemplo de política que el administrador debe actualizar sobre los usuarios o roles de IAM que crean recursos: AWS Glue

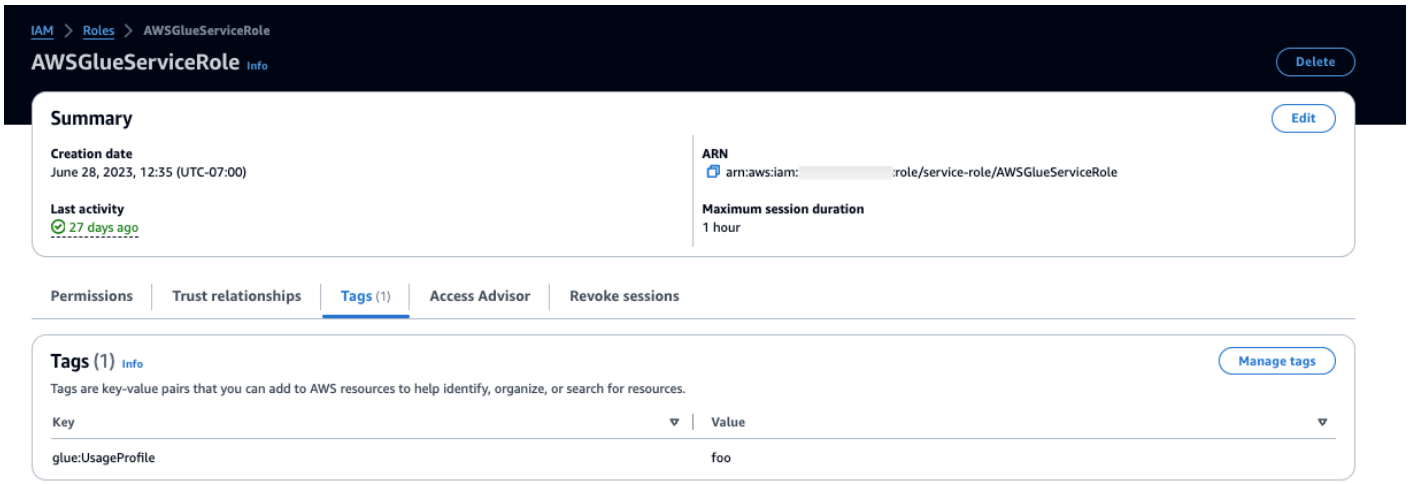
```
{
    "Effect": "Allow",
    "Action": [
        "glue:GetUsageProfile"
```

```
    ],
    "Resource": [
        "arn:aws:glue:us-east-1:123456789012:usageProfile/foo"
    ]
}
```

AWS Glue valida las solicitudes de trabajo, ejecución y sesión en función de los valores especificados en el perfil de AWS Glue uso y genera una excepción si no se permite la solicitud. En el caso de las API sincrónicas, se mostrará un error al usuario. En el caso de las rutas asíncronas, se crea una ejecución de trabajo fallida con el mensaje de error que indica que el parámetro de entrada está fuera del rango permitido para el perfil asignado del usuario o rol.

Para asignar un perfil de uso a un usuario o rol:

1. Abra la consola de IAM (Identity and Access Management).
2. En el menú de navegación de la izquierda, elija Usuarios o Funciones.
3. Elija un usuario o un rol.
4. Elija la pestaña Etiquetas.
5. Elija Añadir nueva etiqueta
6. Añada una etiqueta con la clave `glue:UsageProfile` y el valor del nombre de su perfil de uso.
7. Seleccione Save changes (Guardar cambios)



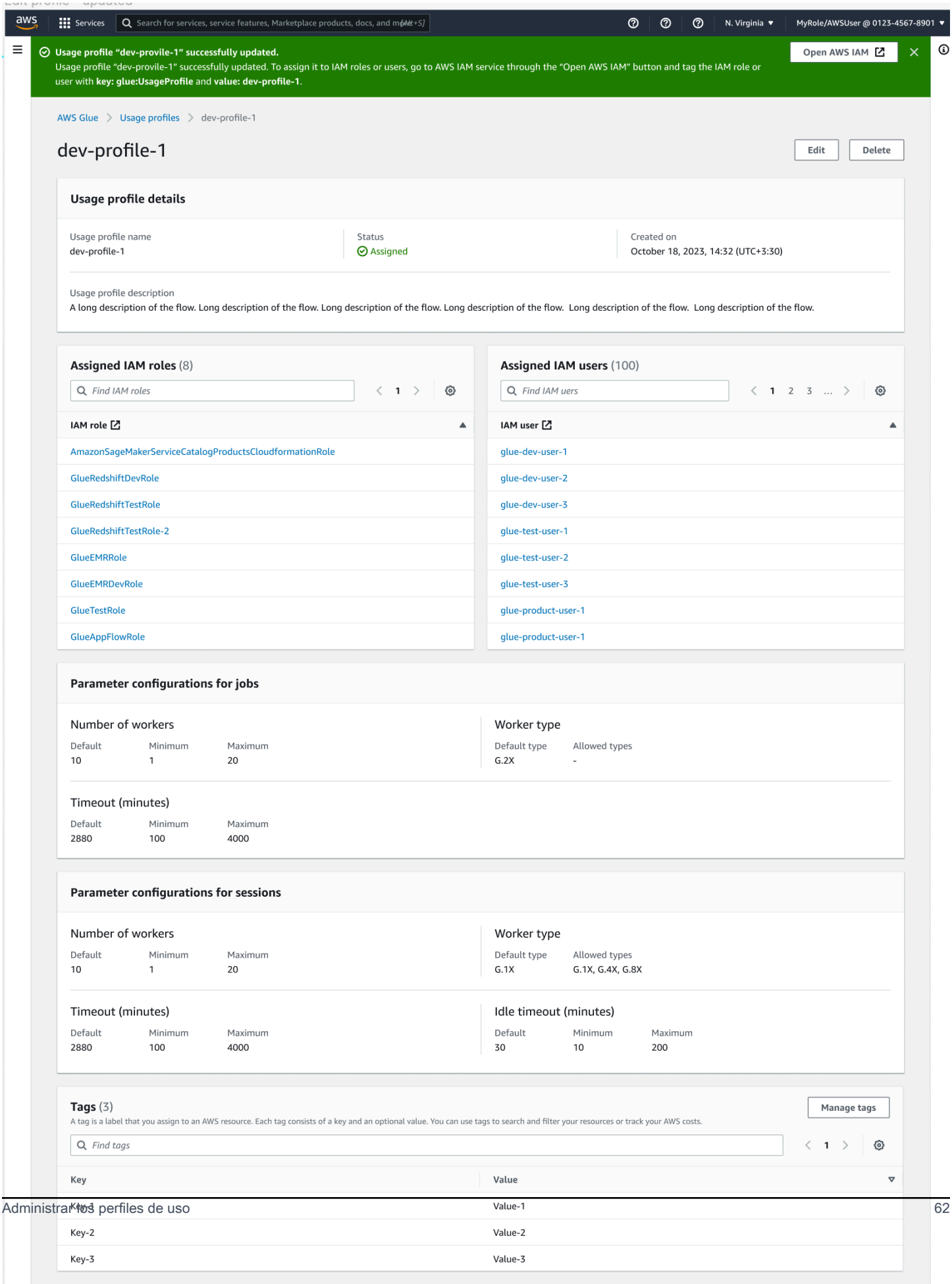
Ver el perfil de uso asignado

Los usuarios pueden ver sus perfiles de uso asignados y usarlos al realizar llamadas a la API para crear recursos de AWS Glue trabajo y sesión, o al iniciar un trabajo.

Los permisos de perfil se proporcionan en las políticas de IAM. Siempre que la política de llamadas tenga el `glue:UsageProfile` permiso, el usuario puede ver el perfil. De lo contrario, aparecerá un error de acceso denegado.

Para ver un perfil de uso asignado:

1. En el menú de navegación de la izquierda, selecciona Gestión de costes.
2. Elija un perfil de uso para el que tenga permisos de visualización.


Services
Search for services, service features, Marketplace products, docs, and more [Alt+S]
N. Virginia
MyRole/AWSUser @ 0123-4567-8901
Usage profile "dev-provile-1" successfully updated.
Usage profile "dev-provile-1" successfully updated. To assign it to IAM roles or users, go to AWS IAM service through the "Open AWS IAM" button and tag the IAM role or user with key: glue:UsageProfile and value: dev-profile-1.
Open AWS IAM
AWS Glue > Usage profiles > dev-profile-1
dev-profile-1
Edit
Delete
Usage profile details
Usage profile name
dev-profile-1
Status
Assigned
Created on
October 18, 2023, 14:32 (UTC+3:30)
Usage profile description
A long description of the flow. Long description of the flow. Long description of the flow. Long description of the flow. Long description of the flow. Long description of the flow.
Assigned IAM roles (8)
Find IAM roles
IAM role
AmazonSageMakerServiceCatalogProductsCloudformationRole
GlueRedshiftDevRole
GlueRedshiftTestRole
GlueRedshiftTestRole-2
GlueEMRRole
GlueEMRDevRole
GlueTestRole
GlueAppFlowRole
Assigned IAM users (100)
Find IAM uers
IAM user
glue-dev-user-1
glue-dev-user-2
glue-dev-user-3
glue-test-user-1
glue-test-user-2
glue-test-user-3
glue-product-user-1
glue-product-user-1
Parameter configurations for jobs
Number of workers
Default 10
Minimum 1
Maximum 20
Worker type
Default type G.2X
Allowed types -
Timeout (minutes)
Default 2880
Minimum 100
Maximum 4000
Parameter configurations for sessions
Number of workers
Default 10
Minimum 1
Maximum 20
Worker type
Default type G.1X
Allowed types G.1X, G.4X, G.8X
Timeout (minutes)
Default 2880
Minimum 100
Maximum 4000
Idle timeout (minutes)
Default 30
Minimum 10
Maximum 200
Tags (3)
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.
Manage tags
Find tags
Key
Value
Key-1
Value-1
Key-2
Value-2
Key-3
Value-3

Perfiles de uso y trabajos

Creación de trabajos con perfiles de uso

Al crear trabajos, se aplicarán los límites y los valores predeterminados establecidos en su perfil de uso. Su perfil se asignará al trabajo al guardarlo.

Ejecutar trabajos con perfiles de uso

Cuando empiezas a ejecutar un trabajo, AWS Glue aplica los límites establecidos en el perfil de la persona que llama. Si no hay una persona que llame directamente, Glue aplicará los límites del perfil asignado al trabajo por su autor.

> **Note**
>
> Cuando un trabajo se ejecute según un cronograma (mediante AWS Glue flujos de trabajo o AWS Glue activadores), el autor aplicará el perfil asignado al trabajo.
> Cuando un servicio externo (Step Functions, MWAA) o una `StartJobRun` API ejecuta un trabajo, se aplicará el límite del perfil de la persona que llama.

En el caso de los AWS Glue flujos de trabajo o los AWS Glue desencadenantes: los trabajos preexistentes deben actualizarse para guardar el nombre del nuevo perfil, de modo que los límites del perfil (número mínimo, máximo y número de trabajadores permitidos) se apliquen durante el tiempo de ejecución para las ejecuciones programadas.

Ver un perfil de uso asignado a los trabajos

Para ver el perfil asignado a sus trabajos (que se utilizará en tiempo de ejecución con AWS Glue flujos de trabajo programados o AWS Glue activadores), consulte la pestaña Detalles del trabajo. También puede consultar el perfil utilizado en las ejecuciones anteriores en la pestaña de detalles de las ejecuciones de tareas.

Actualizar o eliminar un perfil de uso adjunto a un trabajo

El perfil asignado a un trabajo se modifica al actualizarse. Si al autor no se le asigna un perfil de uso, se eliminará cualquier perfil previamente adjunto al trabajo.

Introducción al AWS Glue Data Catalog

El AWS Glue Data Catalog es su almacén de metadatos técnicos persistentes. Se trata de un servicio administrado que puede usar para almacenar, comentar y compartir metadatos en la nube de AWS. Para obtener más información, consulte AWS Glue Data Catalog.

La consola de AWS Glue, junto con algunas interfaces de usuario, se actualizaron recientemente.

Información general

Puede usar este tutorial para crear el primer Catálogo de datos de AWS Glue, que utiliza un bucket de Simple Storage Service (Amazon S3) como origen de datos.

En este tutorial, utilizará la consola de AWS Glue para hacer lo siguiente:

1. Crear una base de datos
2. Crear una tabla
3. Utilizar un bucket de Amazon S3 como origen de datos

Tras completar estos pasos, habrá utilizado correctamente un bucket de Simple Storage Service (Amazon S3) como origen de datos para rellenar el Catálogo de datos de AWS Glue.

Paso 1: Crear una base de datos

Para comenzar, inicie sesión en la AWS Management Console y abra la consola de AWS Glue.

Para crear una base de datos mediante la consola de AWS Glue:

1. En la consola de AWS Glue, elija Databases (Bases de datos) en Data catalog (Catálogo de datos) en el menú de la izquierda.
2. Seleccione Agregar una base de datos.
3. En la página Crear base de datos, ingrese el nombre de la base de datos. En la sección Ubicación: opcional, establezca la ubicación de la URI para que la usen los clientes del catálogo de datos. Si no sabe esto, puede continuar con la creación de la base de datos.
4. (Opcional). Ingrese la descripción de la base de datos.
5. Elija Crear base de datos.

Enhorabuena, acaba de configurar su primera base de datos con la consola de AWS Glue. La nueva base de datos aparecerá en la lista de bases de datos disponibles. Puede editar la base de datos eligiendo su nombre en el panel Databases (Bases de datos).

Pasos siguientes

Otras formas de crear una base de datos:

Acaba de crear una base de datos con la consola de AWS Glue, pero hay otras formas de crear una base de datos:

- Puede utilizar rastreadores para crear una base de datos y tablas automáticamente. Para configurar una base de datos mediante rastreadores, consulte Trabajo con rastreadores en la consola de AWS Glue.
- Puede usar plantillas de AWS CloudFormation. Consulte Crear recursos de AWS Glue con plantillas de AWS Glue Data Catalog.
- También puede crear una base de datos mediante las operaciones de la API de base de datos de AWS Glue.

  Para crear una base de datos mediante la operación `create`, se debe estructurar la solicitud mediante la inclusión de los parámetros (obligatorios) de `DatabaseInput`.

  Por ejemplo:

  A continuación, se muestran ejemplos de cómo puede utilizar CLI, Boto3 o DDL para definir una tabla basada en el mismo archivo flights_data.csv del bucket de S3 que utilizó en el tutorial.

  CLI

  ```
  aws glue create-database --database-input "{\"Name\":\"clidb\"}"
  ```

  Boto3

  ```
  glueClient = boto3.client('glue')

  response = glueClient.create_database(
      DatabaseInput={
          'Name': 'boto3db'
      }
  ```

```
)
```

Para obtener más información sobre los tipos de datos, la estructura y las operaciones de la API de la base de datos, consulte API de la base de datos.

Pasos siguientes

En la siguiente sección, creará una tabla y la agregará a la base de datos.

También puede explorar la configuración y los permisos de su Catálogo de datos. Consulte Trabajo con la configuración del Catálogo de datos en la consola de AWS Glue.

Paso 2. Creación de una tabla

En este paso, utilice la consola de AWS Glue para crear una tabla.

1. En la consola de AWS Glue, elija Tables (Tablas) en el menú de la izquierda.
2. Elija Add table (Añadir tabla).
3. Configure las propiedades de la tabla ingresando un nombre para ella en Table details (Detalles de la tabla).
4. En la sección Databases (Bases de datos), elija en el menú desplegable la base de datos que creó en el paso 1.
5. En la sección Add a data store (Agregar un almacén de datos), S3 estará seleccionado de manera predeterminada como tipo de origen.
6. En la sección Data is located in (Los datos se encuentran en), elija Specified path in another account (Ruta especificada en otra cuenta).
7. Copie y pegue la ruta del campo de entrada Include path (Incluir ruta):

   ```
   s3://crawler-public-us-west-2/flight/2016/csv/
   ```
8. En la sección Data format (Formato de datos), para Classification (Clasificación), elija CSV, y para Delimiter (Delimitador), comma (,) (coma [,]). Elija Siguiente.
9. Tendrá que definir un esquema. Un esquema define la estructura y el formato de un registro de datos. Elija Add column (Agregar columna). (Para obtener más información, consulte Schema registries [Registros de esquemas]).
10. Especifique las propiedades de columna:

a. Ingrese un nombre de columna.

b. Para Column type (Tipo de columna), 'cadena' ya está seleccionada de forma predeterminada.

c. Para Column number (Número de columna), '1' ya está seleccionado de forma predeterminada.

d. Elija Añadir.

11. Tendrá que agregar índices de partición. Es opcional. Para omitir este paso, elija Next (Siguiente).
12. Se muestra un resumen de las propiedades de la tabla. Si todo está correcto, seleccione Crear. En caso contrario, elija Back (Volver) y edite según sea necesario.

Enhorabuena, ha creado correctamente una tabla de forma manual y la ha asociado a una base de datos. La tabla recién creada aparecerá en el panel de tablas. En el panel, puede administrar y modificar todas sus tablas.

Para obtener más información, consulte Trabajo con tablas en la consola de AWS Glue.

Siguientes pasos

Pasos siguientes

Ahora que hemos rellenado el Catálogo de datos, puede comenzar a crear trabajos en AWS Glue. Consulte Creación de trabajos visuales de ETL con AWS Glue Studio.

Además de utilizar la consola, hay otras formas de definir tablas en el Catálogo de datos, entre ellas:

- Creating and running a crawler (Creación y puesta en marcha de un rastreador)
- Adición de clasificadores a un rastreador en AWS Glue
- Uso de la API de tabla de AWS Glue
- Uso de la plantilla de AWS Glue Data Catalog
- Migrating an Apache Hive metastore (Migración de un meta-almacén de Apache Hive)
- Uso de AWS CLI, Boto3 o lenguaje de definición de datos (DDL)

  A continuación, se muestran ejemplos de cómo puede utilizar CLI, Boto3 o DDL para definir una tabla basada en el mismo archivo flights_data.csv del bucket de S3 que utilizó en el tutorial.

Consulte la documentación sobre cómo estructurar un comando AWS CLI. El ejemplo de CLI contiene la sintaxis JSON para el valor “aws glue create-table --table-input”.

CLI

```
{
        "Name": "flights_data_cli",
        "StorageDescriptor": {
            "Columns": [
                {
                    "Name": "year",
                    "Type": "bigint"
                },
                {
                    "Name": "quarter",
                    "Type": "bigint"
                }
            ],
            "Location": "s3://crawler-public-us-west-2/flight/2016/csv",
            "InputFormat": "org.apache.hadoop.mapred.TextInputFormat",
            "OutputFormat":
 "org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat",
            "Compressed": false,
            "NumberOfBuckets": -1,
            "SerdeInfo": {
                "SerializationLibrary":
 "org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe",
                "Parameters": {
                    "field.delim": ",",
                    "serialization.format": ","
                }
            }
        },
        "PartitionKeys": [
            {
                "Name": "mon",
                "Type": "string"
            }
        ],
        "TableType": "EXTERNAL_TABLE",
        "Parameters": {
            "EXTERNAL": "TRUE",
            "classification": "csv",
```

```
            "columnsOrdered": "true",
            "compressionType": "none",
            "delimiter": ",",
            "skip.header.line.count": "1",
            "typeOfData": "file"
        }
    }
```

Boto3

```
import boto3

glue_client = boto3.client("glue")

response = glue_client.create_table(
    DatabaseName='sampledb',
    TableInput={
        'Name': 'flights_data_manual',
    'StorageDescriptor': {
      'Columns': [{
        'Name': 'year',
        'Type': 'bigint'
      }, {
        'Name': 'quarter',
        'Type': 'bigint'
      }],
      'Location': 's3://crawler-public-us-west-2/flight/2016/csv',
      'InputFormat': 'org.apache.hadoop.mapred.TextInputFormat',
      'OutputFormat':
 'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat',
      'Compressed': False,
      'NumberOfBuckets': -1,
      'SerdeInfo': {
        'SerializationLibrary':
 'org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe',
        'Parameters': {
          'field.delim': ',',
          'serialization.format': ','
        }
      },
    },
    'PartitionKeys': [{
```

```
      'Name': 'mon',
      'Type': 'string'
    }],
    'TableType': 'EXTERNAL_TABLE',
    'Parameters': {
      'EXTERNAL': 'TRUE',
      'classification': 'csv',
      'columnsOrdered': 'true',
      'compressionType': 'none',
      'delimiter': ',',
      'skip.header.line.count': '1',
      'typeOfData': 'file'
    }
    }
)
```

DDL

```
CREATE EXTERNAL TABLE `sampledb`.`flights_data` (
  `year` bigint,
  `quarter` bigint)
PARTITIONED BY (
  `mon` string)
ROW FORMAT DELIMITED
  FIELDS TERMINATED BY ','
STORED AS INPUTFORMAT
  'org.apache.hadoop.mapred.TextInputFormat'
OUTPUTFORMAT
  'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION
  's3://crawler-public-us-west-2/flight/2016/csv/'
TBLPROPERTIES (
  'classification'='csv',
  'columnsOrdered'='true',
  'compressionType'='none',
  'delimiter'=',',
  'skip.header.line.count'='1',
  'typeOfData'='file')
```

Configuración del acceso de red a los almacenes de datos

Para ejecutar sus trabajos de extracción, transformación y carga (ETL), AWS Glue debe tener la posibilidad de obtener acceso a los almacenes de datos. Si un trabajo no necesita ejecutarse en la subred de la nube privada virtual (VPC); por ejemplo, transformar datos de Amazon S3 a Amazon S3, no se necesita una configuración adicional.

Si un trabajo debe ejecutarse en la subred de VPC, por ejemplo, deben transformarse datos de un almacén de datos de JDBC en una subred privada, AWS Glue configura interfaces de red elásticas que permiten que sus trabajos se conecten de forma segura con otros recursos dentro de la VPC. A cada interfaz de red elástica se le asigna una dirección IP privada del intervalo de direcciones IP de la subred especificada. No se asignan direcciones IP públicas. Los grupos de seguridad especificados en la conexión AWS Glue se aplican en cada una de las interfaces de red elásticas. Para obtener más información, consulte Configuración de una VPC de Amazon para conexiones JDBC a los almacenes de datos de Amazon RDS de AWS Glue.

Todos los almacenes de datos JDBC a los obtiene acceso el flujo de trabajo deben estar disponibles a partir de la subred VPC. Para obtener acceso a Amazon S3 desde su VPC, se requiere un punto de enlace de la VPC. Si su trabajo necesita obtener acceso a los recursos de la VPC y a la red pública de Internet, la VPC debe tener una gateway de NAT (traducción de direcciones de red) dentro de la VPC.

Un flujo de trabajo o punto de enlace de desarrollo solo puede obtener acceso a una VPC (y subred) a la vez. Si necesita obtener acceso a almacenes de datos en diferentes VPC, dispone de las siguientes opciones:

- Utilice las interconexiones de la VPC para obtener acceso a los almacenes de datos. Para obtener más información acerca de las interconexiones de VPC, consulte Conceptos básicos de las interconexiones de VPC
- Utilice un bucket de Amazon S3 como ubicación de almacenamiento intermediaria. Divida el flujo de trabajo en dos trabajos, con la salida de Amazon S3 del trabajo 1 como entrada del trabajo 2.

Para obtener más información sobre cómo conectarse a un almacén de datos de Amazon Redshift a través de Amazon VPC, consulte the section called “Configurar Redshift”.

Para obtener más información sobre cómo conectarse a los almacenes de datos de Amazon RDS a través de Amazon VPC, consulte the section called “Configuración de una VPC de Amazon para conectarse a los almacenes de datos de Amazon RDS”.

Una vez establecidas las reglas necesarias en Amazon VPC, debe crear una conexión en AWS Glue con las propiedades necesarias para conectarse a sus almacenes de datos. Para obtener más información sobre la conexión, consulte Conexión a datos.

> **Note**
>
> Asegúrese de que configure su entorno de DNS para AWS Glue. Para obtener más información, consulte Configuración de DNS en la VPC.

Temas

- Configuración de una VPC para conectarse a PyPI para AWS Glue
- Configuración de DNS en la VPC

Configuración de una VPC para conectarse a PyPI para AWS Glue

El Python Package Index (PyPI) es un repositorio de software para el lenguaje de programación Python. En este tema se abordan los detalles necesarios para admitir el uso de paquetes instalados por pip (según lo especificado por el creador de la sesión mediante la marca `--additional-python-modules`).

El uso de sesiones AWS Glue interactivas con un conector implica el uso de la red de VPC a través de la subred especificada para el conector. Por lo tanto, los servicios de AWS y otros destinos de red no están disponibles a menos que se establezca una configuración especial.

Las soluciones a este problema incluyen:

- Uso de una puerta de enlace de Internet a la que pueda acceder su sesión.
- Configuración y uso de un bucket de S3 con un repositorio PyPI/Simple que contiene el cierre transitivo de las dependencias de un conjunto de paquetes.
- Uso de un repositorio CodeArtifact que refleje PyPI y esté conectado a su VPC.

Configuración de una puerta de enlace de Internet

Los aspectos técnicos se detallan en los Casos de uso de la puerta de enlace NAT, pero tenga en cuenta estos requisitos al utilizar `--additional-python-modules`. En concreto, `--`

`additional-python-modules` requiere acceso a pypi.org, que viene determinado por la configuración de su VPC. Tenga en cuenta los siguientes requisitos:

1. El requisito de instalar módulos de Python adicionales mediante pip install para la sesión de un usuario. Si la sesión usa un conector, su configuración puede verse afectada.
2. Cuando se utiliza un conector con `--additional-python-modules`, al iniciar la sesión, la subred asociada al `PhysicalConnectionRequirements` del conector debe proporcionar una ruta de red para llegar a pypi.org.
3. Debe determinar si su configuración es correcta o no.

Configuración de un bucket de Amazon S3 para alojar un repositorio de PyPI/Simple de destino

En este ejemplo, se configura una réplica de PyPI en Amazon S3 para un conjunto de paquetes y sus dependencias.

Para configurar la réplica PyPI para un conjunto de paquetes:

```
# pip download all the dependencies
pip download -d s3pypi --only-binary :all: plotly gglplot
pip download -d s3pypi --platform manylinux_2_17_x86_64 --only-binary :all: psycopg2-
binary
# create and upload the pypi/simple index and wheel files to the s3 bucket
s3pypi -b test-domain-name --put-root-index -v s3pypi/*
```

Si ya tienes un repositorio de artefactos existente, tendrá una URL de índice para que pip la use y que podrás proporcionar en lugar de la URL de ejemplo para el bucket de Amazon S3, como se indica anteriormente.

Para usar la URL de índice personalizada, con algunos paquetes de ejemplo:

```
%%configure
{
    "--additional-python-modules": "psycopg2_binary==2.9.5",
    "python-modules-installer-option": "--no-cache-dir --verbose --index-url https://
test-domain-name.s3.amazonaws.com/ --trusted-host test-domain-name.s3.amazonaws.com"
}
```

Configuración de una réplica CodeArtifact de pypi conectada a su VPC

Para configurar un espejo:

1. Cree un repositorio en la misma región que la subred utilizada por el conector.

   Seleccione `Public upstream repositories` y elija `pypi-store`.
2. Proporcione acceso al repositorio desde la VPC de la subred.
3. Especifique la correcta `--index-url` mediante `python-modules-installer-option`.

```
%%configure
{
    "--additional-python-modules": "psycopg2_binary==2.9.5",
    "python-modules-installer-option": "--no-cache-dir --verbose --index-url https://
test-domain-name.s3.amazonaws.com/ --trusted-host test-domain-name.s3.amazonaws.com"
}
```

Para obtener más información, consulte Usar CodeArtifact desde una VPC.

Configuración de DNS en la VPC

El sistema de nombres de dominio (DNS) es un estándar mediante el cual los nombres utilizados en Internet se resuelven a sus direcciones IP correspondientes. Un nombre de host de DNS designa a un equipo de forma exclusiva y se compone de un nombre de host y un nombre de dominio. Los servidores DNS resuelven los nombres de host DNS a sus direcciones IP correspondientes.

Para configurar DNS en la VPC, asegúrese de que tanto los nombres de host de DNS, como la resolución de DNS están habilitados en la VPC. Los atributos de red de VPC `enableDnsHostnames` y `enableDnsSupport` deben establecerse en `true`. Para ver y modificar estos atributos, vaya a la consola de la VPC en https://console.aws.amazon.com/vpc/.

Para obtener más información, consulte Utilización de DNS con su VPC. Además, puede utilizar AWS CLI y llamar al comando modify-vpc-attribute para configurar los atributos de red de VPC.

> **Note**
>
> Si utiliza Route 53, confirme que la configuración no sustituye los atributos de red de DNS.

Configuración del cifrado en AWS Glue

En el siguiente flujo de trabajo de ejemplo se resaltan las opciones que se deben configurar al utilizar el cifrado con AWS Glue. El ejemplo ilustra el uso de claves de AWS Key Management Service (AWS KMS) específicas, pero puede elegir otras configuraciones en función de sus necesidades particulares. Este flujo de trabajo destaca solo las opciones que atañen al cifrado a la hora de configurar AWS Glue.

1. Si el usuario de la consola de AWS Glue no utiliza una política de permisos que permita todas las operaciones de la API de AWS Glue (por ejemplo, `"glue:*"`), confirme que las siguientes acciones están permitidas:
   - `"glue:GetDataCatalogEncryptionSettings"`
   - `"glue:PutDataCatalogEncryptionSettings"`
   - `"glue:CreateSecurityConfiguration"`
   - `"glue:GetSecurityConfiguration"`
   - `"glue:GetSecurityConfigurations"`
   - `"glue:DeleteSecurityConfiguration"`
2. Cualquier cliente que obtenga acceso a un catálogo de cifrado o escriba en él, es decir, cualquier usuario de la consola, rastreador, trabajo o punto de enlace de desarrollo, necesitará los siguientes permisos.

```
{
 "Version": "2012-10-17",
  "Statement": {
     "Effect": "Allow",
     "Action": [
           "kms:GenerateDataKey",
           "kms:Decrypt",
           "kms:Encrypt"
      ],
     "Resource": "<key-arns-used-for-data-catalog>"
   }
}
```

3. Cualquier usuario o rol que obtenga acceso a una contraseña de conexión cifrada necesitará los siguientes permisos:

```
{
```

```
 "Version": "2012-10-17",
  "Statement": {
     "Effect": "Allow",
     "Action": [
           "kms:Decrypt"
          ],
     "Resource": "<key-arns-used-for-password-encryption>"
          }
}
```

4. El rol de cualquier trabajo de extracción, transformación y carga (ETL) que escriba datos cifrados en Amazon S3 necesitará los siguientes permisos.

```
{
 "Version": "2012-10-17",
  "Statement": {
     "Effect": "Allow",
     "Action": [
           "kms:Decrypt",
           "kms:Encrypt",
           "kms:GenerateDataKey"
      ],
     "Resource": "<key-arns-used-for-s3>"
   }
}
```

5. Cualquier rastreador o trabajo de ETL que escriba Registros de Amazon CloudWatch cifrados necesitará los siguientes permisos en la política de claves y las políticas de IAM.

   En la política de claves (no en la política de IAM):

```
{
  "Effect": "Allow",
  "Principal": {
   "Service": "logs.region.amazonaws.com"
  },
  "Action": [
   "kms:Encrypt*",
   "kms:Decrypt*",
   "kms:ReEncrypt*",
   "kms:GenerateDataKey*",
   "kms:Describe*"
  ],
```

```
  "Resource": "<arn of key used for ETL/crawler cloudwatch encryption>"
 }
```

Para obtener más información acerca de las políticas de claves, consulte Uso de las políticas de claves en AWS KMS en la Guía para desarrolladores de AWS Key Management Service.

En la política de IAM, adjunte el permiso de `logs:AssociateKmsKey`:

```
{
  "Effect": "Allow",
  "Principal": {
   "Service": "logs.region.amazonaws.com"
  },
  "Action": [
   "logs:AssociateKmsKey"
  ],
  "Resource": "<arn of key used for ETL/crawler cloudwatch encryption>"
 }
```

6. Cualquier flujo de trabajo de ETL que utilice un marcador de trabajo cifrado necesitará los siguientes permisos:

```
{
 "Version": "2012-10-17",
  "Statement": {
     "Effect": "Allow",
     "Action": [
           "kms:Decrypt",
           "kms:Encrypt"
      ],
     "Resource": "<key-arns-used-for-job-bookmark-encryption>"
   }
}
```

7. En la consola de AWS Glue, elija Settings (Configuración) en el panel de navegación.
   a. En la página Data catalog settings (Configuración del Data Catalog), cifre su Data Catalog al seleccionar Metadata encryption (Cifrado de metadatos). Esta opción cifra todos los objetos del Data Catalog con la clave AWS KMS que usted elija.
   b. Para AWS KMS key (Clave KMS), elija aws/glue. También puede elegir una clave AWS KMS que haya creado.

> **Important**
>
> AWS Glue solo soporta claves maestras de cliente (CMK) simétricas. La lista de AWS KMS key (Clave KMS) muestra únicamente claves simétricas. Sin embargo, si selecciona Choose a AWS KMS key ARN (Elegir un ARN de clave KMS), la consola le permite introducir un ARN para cualquier tipo de clave. Asegúrese de introducir sólo ARN para claves simétricas.

Cuando el cifrado está habilitado, el cliente que accede a Data Catalog debe tener permisos de AWS KMS.

8. En el panel de navegación, seleccione Security configurations (Configuraciones de seguridad). Una configuración de seguridad es un conjunto de propiedades de seguridad que se pueden usar para configurar procesos de AWS Glue. A continuación, elija Add security configuration (Agregar configuración de seguridad). En la configuración, elija cualquiera de las siguientes opciones:

   a. Seleccione S3 encryption (Cifrado de S3). Para Encryption mode (Modo de cifrado), seleccione SSE-KMS. Para la AWS KMS key (Clave de), elija aws/s3 (asegúrese de que el usuario tiene permiso para usar esta clave). Esto permite a los datos escritos por el trabajo en Amazon S3, utilizar la clave AWS KMS de AWS Glue administrada por AWS.

   b. Seleccione CloudWatch logs encryption (Cifrado de registros de Cloudwatch) y una CMK. (Asegúrese de que el usuario tenga permiso para usar esta clave). Para obtener más información, consulte Cifrar datos de registro en CloudWatch Logs mediante AWS KMS en la Guía para desarrolladores de AWS Key Management Service.

      > **Important**
      >
      > AWS Glue solo soporta claves maestras de cliente (CMK) simétricas. La lista de AWS KMS key (Clave KMS) muestra únicamente claves simétricas. Sin embargo, si selecciona Choose a AWS KMS key ARN (Elegir un ARN de clave KMS), la consola le permite introducir un ARN para cualquier tipo de clave. Asegúrese de introducir sólo ARN para claves simétricas.

   c. Seleccione Advanced properties (Propiedades avanzadas) y Job bookmark encryption (Cifrado de marcador de trabajos). Para la AWS KMS key (Clave de), elija aws/glue (asegúrese de que el usuario tiene permiso para usar esta clave). Esto permite el cifrado de marcadores de trabajo escritos en Amazon S3 con la clave AWS KMS de AWS Glue.

9. En el panel de navegación, elija Connections (Conexiones).
   a. Seleccione Add connection (Agregar conexión) para crear una conexión al almacén de datos de Java Database Connectivity (JDBC) que es el destino de su flujo de trabajo de ETL.
   b. Para forzar el uso del cifrado de Capa de conexión segura (SSL), seleccione Require SSL connection (Exigir conexión SSL) y pruebe la conexión.
10. En el panel de navegación, seleccione Trabajos.
   a. Seleccione Add job (Añadir trabajo) para crear una flujo de trabajo que transforma los datos.
   b. En la definición de trabajo, elija la configuración de seguridad que ha creado.
11. En la consola de AWS Glue, ejecute el flujo de trabajo bajo demanda. Compruebe que todos los datos de Amazon S3 escritos por el trabajo, los CloudWatch Logs escritos por el trabajo y los marcadores de trabajo están cifrados.

Configuración de redes para el desarrollo de AWS Glue

Para ejecutar sus scripts del servicio ETL (extracción, transformación y carga) con AWS Glue, puede desarrollar y probar sus scripts mediante un punto de conexión de desarrollo. Los trabajos de la versión 2.0 de AWS Glue no soportan el uso de puntos de enlace desarrollo. Para las versiones 2.0 y posteriores, el método de desarrollo preferido es utilizar el bloc de notas de Jupyter con uno de los kernel de AWS Glue. Para obtener más información, consulte the section called “Introducción a las sesiones interactivas de AWS Glue”.

Configuración de su red para un punto de conexión de desarrollo

Cuando configura un punto de enlace de desarrollo, debe especificar una nube virtual privada (VPC), una subred y los grupos de seguridad.

> **Note**
>
> Asegúrese de que configure su entorno de DNS para AWS Glue. Para obtener más información, consulte Configuración de DNS en la VPC.

Para que AWS Glue pueda obtener acceso a los recursos necesarios, agregue una fila a su tabla de ruta de la subred para asociar una lista de prefijos para Amazon S3 al punto de enlace de la VPC. Se necesita un ID de lista de prefijo para crear una regla de grupo de seguridad saliente que permita

al tráfico de una VPC obtener acceso a un servicio de AWS a través de un punto de conexión de la VPC. Para facilitar la conexión a un servidor de blocs de notas que esté asociado a este punto de enlace de desarrollo, en el equipo local agregue una fila a la tabla de ruta para incorporar un ID de gateway de Internet. Para obtener más información, consulte Puntos de conexión de la VPC. Actualice las rutas de subred para que sean similares a la siguiente tabla:

Destino	Objetivo		
10.0.0.0/16	local		
pl-id para Amazon S3	vpce-id		
0.0.0.0/0	igw-xxxx		

Para que AWS Glue pueda comunicarse entre sus componentes, especifique un grupo de seguridad con una regla de entrada con autorreferencia para todos los puertos TCP. Si crea una regla de entrada con autorreferencia, puede restringir el origen al mismo grupo de seguridad de la VPC y no está abierto a todas las redes. Puede que el grupo de seguridad predeterminado de la VPC ya tenga una regla de entrada con autorreferencia para ALL Traffic.

Para configurar un grupo de seguridad

1. Inicie sesión en la AWS Management Console y abra la consola de Amazon EC2 en https://console.aws.amazon.com/ec2/.
2. En el panel de navegación izquierdo, elija Security Groups.
3. Elija un grupo de seguridad ya existente en la lista o seleccione Create Security Group (Crear grupo de seguridad) para usarlo con el punto de enlace de desarrollo.
4. En el panel del grupo de seguridad, vaya a la pestaña Inbound (Entrante).
5. Añada una regla con autorreferencia para que los componentes de AWS Glue puedan comunicarse. En concreto, añada o confirme que hay una regla con Type (Tipo) `All TCP`, Protocol (Protocolo) `TCP`, Port Range (Intervalo de puertos) con todos los puertos y el Source (Origen) con el mismo nombre de grupo de seguridad que Group ID (ID de grupo).

   La regla de entrada tiene un aspecto similar al siguiente:

Tipo	Protocolo	Rango de puerto	Origen
Todos los TCP	TCP	0–65535	*security-group*

A continuación se muestra un ejemplo de una regla de entrada con autorreferencia:

6. Añada también una regla para el tráfico saliente. Abra el tráfico saliente a todos los puertos o cree una regla con autorreferencia de Type (Tipo) `All TCP`, Protocol (Protocolo) `TCP`, Port Range (Intervalo de puertos) con todos los puertos y el Source (Origen) con el mismo nombre de grupo de seguridad que Group ID (ID de grupo).

   La regla de salida será similar a una de estas reglas:

Tipo	Protocolo	Rango de puerto	Destino
Todos los TCP	TCP	0–65535	*security-group*
All Traffic	ALL	ALL	0.0.0.0/0

Configuración de Amazon EC2 para el servidor de blocs de notas

Con un punto de conexión de desarrollo, puede crear un servidor de blocs de notas para probar sus scripts de ETL con cuaderno de Jupyter. Para habilitar la comunicación con su bloc de notas, especifique un grupo de seguridad con reglas de entrada tanto para HTTPS (puerto 443) como para SSH (puerto 22). Asegúrese de que el origen de la regla sea 0.0.0.0/0 o la dirección IP del equipo que se va a conectar al bloc de notas.

Para configurar un grupo de seguridad

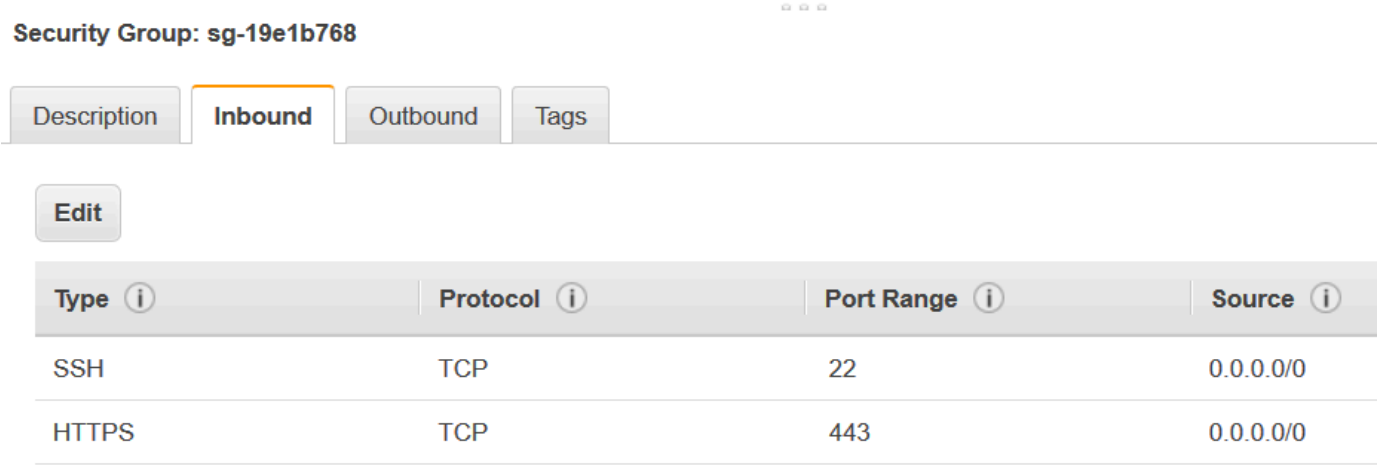
1. Inicie sesión en la AWS Management Console y abra la consola de Amazon EC2 en https://console.aws.amazon.com/ec2/.
2. En el panel de navegación izquierdo, elija Security Groups.
3. Elija un grupo de seguridad ya existente en la lista o seleccione Create Security Group (Crear grupo de seguridad) para usarlo con su servidor de blocs de notas. El grupo de seguridad

asociado con su punto de enlace de desarrollo también se utiliza para crear su servidor de blocs de notas.

4. En el panel del grupo de seguridad, vaya a la pestaña Inbound (Entrante).
5. Añada reglas de entrada similares a la siguiente:

Tipo	Protocolo	Rango de puerto	Origen
SSH	TCP	22	0.0.0.0/0
HTTPS	TCP	443	0.0.0.0/0

A continuación se muestra un ejemplo de las reglas de entrada del grupo de seguridad:



Detección y catalogación de datos en AWS Glue

El AWS Glue Data Catalog es un repositorio centralizado que almacena metadatos sobre los conjuntos de datos de su organización. Actúa como un índice para las métricas de tiempo de ejecución, esquema y ubicación de sus orígenes de datos. Los metadatos se almacenan en tablas de metadatos, en las que cada tabla representa un único almacén de datos.

Para completar el Catálogo de datos, puede usar un rastreador, que escanea automáticamente los orígenes de datos y extrae los metadatos. Los rastreadores pueden conectarse a orígenes de datos internos (basados en AWS) y externos a AWS.

Para obtener más información sobre los orígenes de datos compatibles, consulte ¿Qué almacenes de datos puedo rastrear?.

También puede crear tablas en el Catálogo de datos manualmente, para lo cual debe definir la estructura de la tabla, el esquema y la estructura de particiones de acuerdo con sus requisitos específicos.

Para obtener más información sobre la creación manual de tablas de metadatos, consulte Cómo definir los metadatos manualmente.

Puede usar la información del Catálogo de datos para crear y supervisar sus trabajos de ETL. El Catálogo de datos se integra con otros servicios de análisis de AWS y proporciona una vista unificada de los orígenes de datos, lo que facilita la administración y el análisis de los datos.

- Amazon Athena: Almacene y consulte los metadatos de las tablas en el Catálogo de datos para los datos de Amazon S3 con SQL.
- AWS Lake Formation: Defina y administre las políticas de acceso a los datos detalladas y audite el acceso a los datos de forma centralizada.
- Amazon EMR: Acceda a los orígenes de datos definidos en el Catálogo de datos para el procesamiento de macrodatos.
- Amazon SageMaker: Cree, entrene e implemente modelos de machine learning de forma rápida y segura.

Características clave del Catálogo de datos

A continuación se detallan los aspectos clave del Catálogo de datos.

Repositorio de metadatos

El Catálogo de datos actúa como un repositorio central de metadatos y almacena información sobre la ubicación, el esquema y las propiedades de los orígenes de datos. Estos metadatos se organizan en bases de datos y tablas, de forma similar a un catálogo de una base de datos relacional tradicional.

Detección automática de datos

Los Rastreador de AWS Glue pueden detectar y catalogar los orígenes de datos nuevos o actualizados automáticamente, lo que reduce la sobrecarga de la administración manual de los metadatos y garantiza que su Catálogo de datos permanezca actualizado. Al catalogar sus orígenes de datos, el Catálogo de datos facilita a los usuarios y las aplicaciones la detección y la comprensión de los activos de datos disponibles en su organización, lo que promueve la reutilización de los datos y la colaboración.

El Catálogo de datos es compatible con una amplia gama de orígenes de datos, incluidos Amazon S3, Amazon RDS, Amazon Redshift y Apache Hive, entre otros. Puede inferir y almacenar metadatos de estos orígenes automáticamente mediante el uso de Rastreador de AWS Glue.

Para obtener más información, consulte, Uso de rastreadores para completar el Catálogo de datos .

Administración de esquemas

El Catálogo de datos captura y administra automáticamente el esquema de sus orígenes de datos, incluida la inferencia, la evolución y el control de versiones del esquema. Para actualizar el esquema y las particiones del Catálogo de datos, puede usar los trabajos de ETL de AWS Glue.

Optimización de tablas

Para mejorar el rendimiento de lectura de los servicios de análisis de AWS, como Amazon Athena, Amazon EMR, y los trabajos de ETL de AWS Glue, el Catálogo de datos ofrece una compactación administrada (un proceso que compacta objetos pequeños de Amazon S3 para convertirlos en objetos más grandes) para procesar las tablas de Iceberg del Catálogo de datos. Puede usar la consola de AWS Glue, la consola de AWS Lake Formation, la AWS CLI o la API de AWS para activar o desactivar la compactación de las tablas de Iceberg individuales que están en el Catálogo de datos.

Para obtener más información, consulte Optimización de las tablas de Iceberg.

Estadísticas de las columnas

Puede calcular las estadísticas a nivel de columna para las tablas del Catálogo de datos en formatos de datos como Parquet, ORC, JSON, ION, CSV y XML sin necesidad de configurar canalizaciones de datos adicionales. Las estadísticas de columnas le ayudan a entender los perfiles de datos al obtener información sobre los valores de una columna. El Catálogo de datos permite generar estadísticas para los valores de las columnas, como los valores mínimo y máximo, los valores nulos totales y distintos totales, la longitud media de los valores y el total de apariciones de valores verdaderos.

Para obtener más información, consulte Cómo optimizar el rendimiento de las consultas con las estadísticas de columnas.

Linaje de datos

El Catálogo de datos mantiene un registro de las transformaciones y operaciones realizadas con los datos, y proporciona información sobre el linaje de los datos. Esta información sobre el linaje es valiosa para la auditoría, el cumplimiento y la comprensión de la procedencia de los datos.

Integración con otros servicios de AWS

El Catálogo de datos se integra perfectamente con otros servicios de AWS, como AWS Lake Formation, Amazon Athena, Amazon Redshift Spectrum y Amazon EMR. Esta integración le permite consultar y analizar los datos de varios almacenes de datos mediante el uso de una única capa de metadatos coherente.

Seguridad y control de acceso

AWS Glue se integra con AWS Lake Formation para promover un control minucioso del acceso a los recursos del Catálogo de datos, lo que le permite administrar los permisos y proteger el acceso a sus activos de datos en función de las políticas y los requisitos de su organización. AWS Glue se integra con AWS Key Management Service (AWS KMS) para cifrar los metadatos almacenados en el Catálogo de datos.

Temas

- Cómo completar el Catálogo de datos de AWS Glue
- Cómo completar y administrar las tablas transaccionales
- Administración del Catálogo de datos
- Cómo acceder al Catálogo de datos
- Prácticas recomendadas para el uso del Catálogo de datos de AWS Glue

- AWS Glue Schema Registry

Cómo completar el Catálogo de datos de AWS Glue

Puede completar el AWS Glue Data Catalog mediante los siguientes métodos:

- Rastreador de AWS Glue: El Rastreador de AWS Glue puede detectar y catalogar automáticamente orígenes de datos, como bases de datos, lagos de datos y datos de streaming. Los rastreadores son el método más común y recomendado para completar el Catálogo de datos, ya que pueden detectar e inferir automáticamente los metadatos de una amplia variedad de orígenes de datos.
- Adición manual de metadatos: Puede definir manualmente las bases de datos, las tablas y los detalles de conexión, y agregarlos al Catálogo de datos con la consola de AWS Glue, la consola de Lake Formation, la AWS CLI o las API de AWS Glue. El ingreso manual resulta útil cuando quiere catalogar los orígenes de datos que no se pueden rastrear.
- Integración con otros servicios de AWS: Puede completar el Catálogo de datos con metadatos de servicios como AWS Lake Formation y Amazon Athena. Estos servicios pueden detectar y registrar los orígenes de datos en el Catálogo de datos.
- Adición desde un repositorio de metadatos existente: Si tiene un almacén de metadatos existente, como Apache Hive Metastore, puede usar AWS Glue para importar esos metadatos al Catálogo de datos. Para obtener más información, consulte Migración entre un metaalmacén de Hive y AWS Glue Data Catalog en GitHub.

Temas

- Uso de rastreadores para completar el Catálogo de datos
- Cómo definir los metadatos manualmente
- Integración con otros servicios de AWS
- Configuración del Catálogo de datos

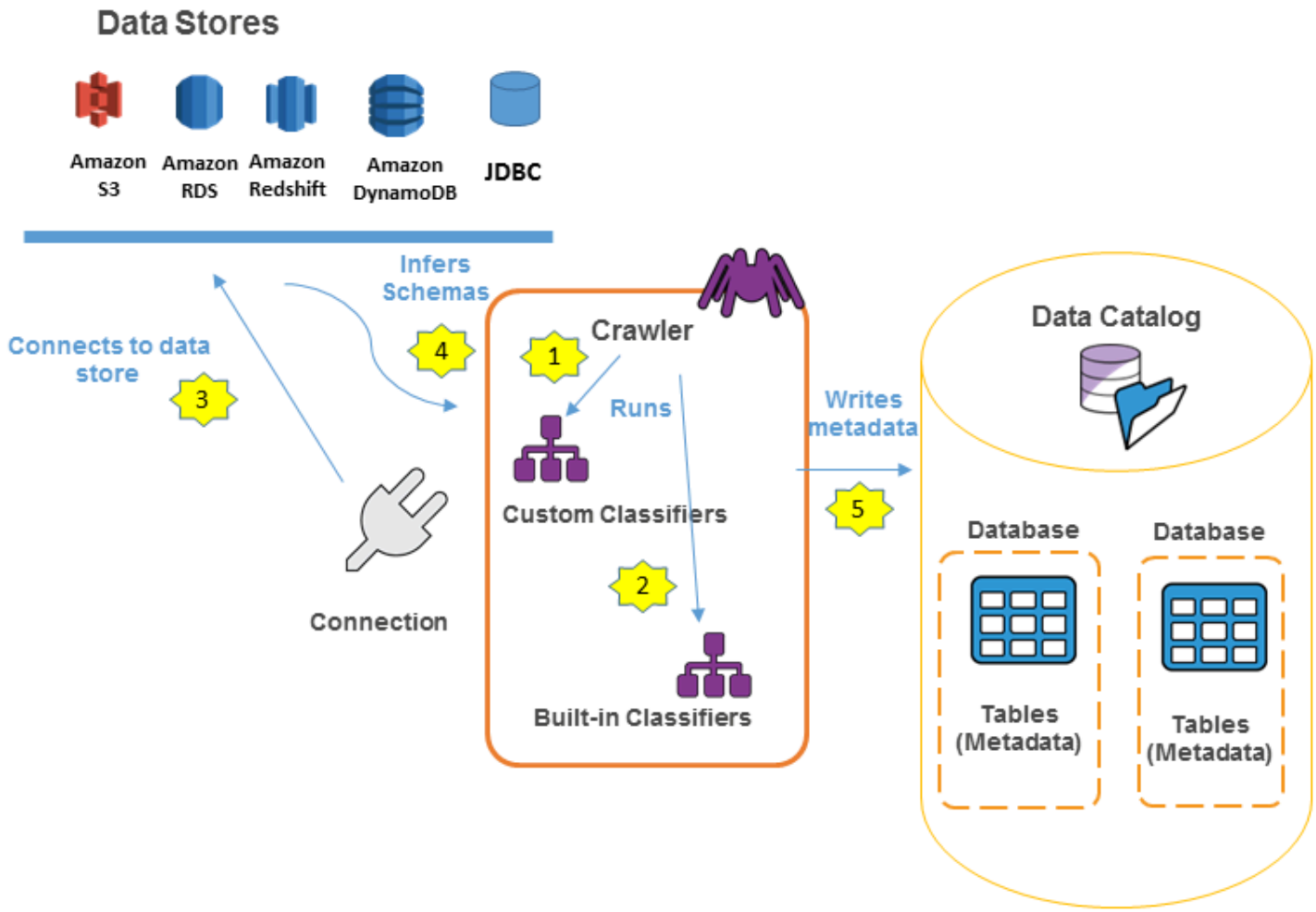
Uso de rastreadores para completar el Catálogo de datos

Puede usar un Rastreador de AWS Glue para completar el AWS Glue Data Catalog con bases de datos y tablas. Este es el método principal usado por la mayoría de los usuarios de AWS Glue. Un rastreador puede rastrear varios almacenes de datos en una única ejecución. Cuando finaliza, el rastreador crea o actualiza una o varias tablas del Catálogo de datos. Los trabajos de extracción,

transformación y carga (ETL) que define en AWS Glue usan estas tablas del Catálogo de datos como orígenes y destinos. El trabajo de ETL lee y escribe en los almacenes de datos que se especifican en las tablas de origen y destino del Catálogo de datos.

Flujo de trabajo

En el siguiente diagrama de flujo de flujo de trabajo se muestra cómo los rastreadores de AWS Glue interactúan con almacenes de datos y otros elementos para rellenar el Catálogo de datos.



Este es el flujo de flujo de trabajo general de rellenado de AWS Glue Data Catalog por parte de un rastreador:

1. Un rastreador ejecuta cualquier clasificador personalizado que elija para inferir el formato y el esquema de sus datos. Debe proporcionar el código para clasificadores personalizados, que se ejecutan en el orden especificado.

El primer clasificador personalizado en reconocer correctamente la estructura de sus datos se usa para crear un esquema. Los clasificadores personalizados que aparecen más abajo en la lista se omiten.

2. Si no coincide ningún clasificador con el esquema de sus datos, los clasificadores integrados intentarán reconocer el esquema de sus datos. Un ejemplo de un clasificador integrado es uno que reconoce JSON.
3. El rastreador se conecta al almacén de datos. Algunos almacenes de datos requieren propiedades de conexión para el acceso del rastreador.
4. El esquema inferido se crea para sus datos.
5. El rastreador escribe los metadatos en el Catálogo de datos. Una definición de tabla contiene metadatos acerca de los datos de su almacén de datos. La tabla se escribe en una base de datos, que es un contenedor de tablas en el Catálogo de datos. Entre los atributos de una tabla se incluye la clasificación, que es una etiqueta creada por el clasificador que determinó el esquema de tabla.

Temas

- Funcionamiento de los rastreadores
- ¿Qué almacenes de datos puedo rastrear?
- ¿Cómo determina un rastreador cuándo crear particiones?
- Requisitos previos del rastreador
- Configuración de rastreadores
- Adición de clasificadores a un rastreador en AWS Glue
- Programación de un rastreador de AWS Glue
- Ver los resultados y detalles del rastreador
- Personalización del comportamiento del rastreador
- Tutorial: agregar un rastreador de AWS Glue

Funcionamiento de los rastreadores

Cuando se ejecuta un rastreador, realiza las siguientes acciones para interrogar a un almacén de datos:

- Clasifica los datos para determinar el formato, el esquema y las propiedades asociadas de los datos sin procesar: puede configurar los resultados de clasificación mediante la creación de un clasificador personalizado.
- Agrupa los datos en tablas o particiones: los datos se agrupan en función de la heurística de rastreador.
- Escribe los metadatos en el Catálogo de datos: puede configurar cómo el rastreador agrega, actualiza y elimina tablas y particiones.

Al definir un rastreador, puede elegir uno o varios clasificadores que evalúen el formato de sus datos para inferir un esquema. Al ejecutarse el rastreador, el primer clasificador de su lista en reconocer correctamente su almacén de datos se usa para crear un esquema para su tabla. Puede usar clasificadores integrados o definir los suyos propios. Puede definir sus clasificadores personalizados en una operación independiente, antes de definir los rastreadores. AWS Glue proporciona clasificadores integrados para inferir esquemas a partir de archivos comunes con formatos entre los que se incluyen JSON, CSV y Apache Avro. Para ver la lista actual de clasificadores integrados en AWS Glue, consulte Clasificadores integrados en AWS Glue.

Las tablas de metadatos que crea un rastreador se incluyen en una base de datos al definir un rastreador. Si su rastreador no especifica una base de datos, sus tablas se colocan en la base de datos predeterminada. Además, cada tabla tiene una columna de clasificación que rellena el clasificador que reconoció correctamente el almacén de datos en primer lugar.

Si se comprime el archivo que se rastrea, el rastreador debe descargarlo para procesarlo. Cuando un rastreador se ejecuta, interroga los archivos para determinar su formato y tipo de compresión, y escribe estas propiedades en el Catálogo de datos. Algunos formatos de archivo (por ejemplo, Apache Parquet) le permiten comprimir partes del archivo a medida que se escribe. Para estos archivos, los datos comprimidos son un componente interno del archivo y AWS Glue no rellena la propiedad `compressionType` cuando escribe tablas en el Catálogo de datos. Por el contrario, si un archivo completo se comprime mediante un algoritmo de compresión (por ejemplo, gzip), la propiedad `compressionType` se rellena cuando las tablas se escriben en el Catálogo de datos.

El rastreador genera los nombres para las tablas que crea. Los nombres de las tablas que se almacenan en el AWS Glue Data Catalog siguen estas reglas:

- Solo se permiten caracteres alfanuméricos y guiones bajos (_).
- Ningún prefijo personalizado puede tener más de 64 caracteres.

- La longitud máxima del nombre no puede ser superior a 128 caracteres. El rastreador trunca nombres generados para que quepan en el límite.
- Si se encuentran nombres de tabla duplicados, el rastreador añade un sufijo de cadena hash al nombre.

Si su rastreador se ejecuta más de una vez, quizás en una programación, busca archivos o tablas nuevos o cambiados en su almacén de datos. La salida del rastreador incluye nuevas tablas y particiones encontradas desde una ejecución anterior.

¿Qué almacenes de datos puedo rastrear?

Los rastreadores pueden rastrear los siguientes almacenes de datos basados en archivos y almacenes de datos basados en tablas.

Tipo de acceso que utiliza el rastreador	Almacenes de datos
Cliente nativo	• Amazon Simple Storage Service (Amazon S3) • Amazon DynamoDB • Delta Lake 2.0.x • Apache Iceberg 1.5 • Apache Hudi 0.14
JDBC	Amazon Redshift Snowflake Dentro de Amazon Relational Database Service (Amazon RDS) o externo a Amazon RDS: • Amazon Aurora • MariaDB • Microsoft SQL Server • MySQL • Oracle • PostgreSQL

Tipo de acceso que utiliza el rastreador	Almacenes de datos
Cliente de MongoDB	• MongoDB • MongoDB Atlas • Amazon DocumentDB (con compatibilidad con MongoDB)

> **Note**
>
> Actualmente, AWS Glue no admite rastreadores para transmisiones de datos.

Para los almacenes de datos JDBC, MongoDB, MongoDB Atlas y Amazon DocumentDB (con compatibilidad con MongoDB), debe especificar una conexión de AWS Glue que el rastreador pueda usar para conectarse al almacén de datos. Para Amazon S3, puede especificar opcionalmente una conexión de tipo Red. Una conexión es un objeto del Catálogo de datos que almacena información de conexión, como credenciales, URL, información de Amazon Virtual Private Cloud, etc. Para obtener más información, consulte Conexión a datos.

A continuación se enumeran las versiones de controladores compatibles con el rastreador:

Producto	Controlador compatible con el rastreador
PostgreSQL	42.2.1
Amazon Aurora	Igual que los controladores de rastreadores nativos
MariaDB	8.0.13
Microsoft SQL Server	6.1.0
MySQL	8.0.13
Oracle	11.2.2
Amazon Redshift	4.1

Producto	Controlador compatible con el rastreador
Snowflake	3.13.20
MongoDB	4.7.2
MongoDB Atlas	4.7.2

A continuación, se muestran notas sobre los distintos almacenes de datos.

Amazon S3

Puede elegir rastrear una ruta en su cuenta o en otra cuenta. Si todos los archivos de Amazon S3 de una carpeta tienen el mismo esquema, el rastreador crea una tabla. Además, si el objeto de Amazon S3 está particionado, solo se crea una tabla de metadatos y se agrega información de particiones al Catálogo de datos de esa tabla.

Amazon S3 y Amazon DynamoDB

Los rastreadores utilizan un rol de AWS Identity and Access Management (IAM) para obtener permiso y acceder a sus almacenes de datos. El rol que se transfiere al rastreador debe tener permiso para obtener acceso a las rutas de Amazon S3 y a las tablas de Amazon DynamoDB que se rastrean.

Amazon DynamoDB

Al definir un rastreador mediante la consola de AWS Glue, especifica una tabla de DynamoDB. Si usa la API de AWS Glue, especifica una lista de tablas. Puede elegir rastrear sólo una pequeña muestra de los datos para reducir los tiempos de ejecución del rastreador.

Delta Lake

En cada almacén de datos de Delta Lake, debe especificar cómo crear tablas de Delta:

- Crear tablas nativas: se permite la integración a los motores de consulta que permiten consultar el registro de transacciones de Delta directamente. Para obtener más información, consulte Consultar las tablas de Delta Lake.
- Crear tablas de enlaces simbólicos: se crea una carpeta de `_symlink_manifest` con los archivos de manifiesto particionados mediante las claves de partición en función de los parámetros de configuración especificados.

Iceberg

Para cada almacén de datos de Iceberg, debe especificar una ruta de Amazon S3 que contenga los metadatos de las tablas de Iceberg. Si el rastreador descubre metadatos de tablas de Iceberg, los registra en el Data Catalog. Puede establecer una programación para que el rastreador mantenga las tablas actualizadas.

Puede definir estos parámetros para el almacén de datos:

- Exclusiones: permite omitir determinadas carpetas.
- Profundidad máxima de recorrido: establece el límite de profundidad que el rastreador puede rastrear en su bucket de Amazon S3. La profundidad de recorrido máxima predeterminada es 10 y la profundidad máxima que puede establecer es 20.

Hudi

Para cada almacén de datos de Hudi, debe especificar una ruta de Amazon S3 que contenga los metadatos de las tablas de Hudi. Si el rastreador descubre metadatos de la tabla de Hudi, los registra en el Data Catalog. Puede establecer una programación para que el rastreador mantenga las tablas actualizadas.

Puede definir estos parámetros para el almacén de datos:

- Exclusiones: permite omitir determinadas carpetas.
- Profundidad máxima de recorrido: establece el límite de profundidad que el rastreador puede rastrear en su bucket de Amazon S3. La profundidad de recorrido máxima predeterminada es 10 y la profundidad máxima que puede establecer es 20.

> **Note**
>
> Las columnas de marcas temporales con tipos lógicos `millis` se interpretarán como `bigint` debido a una incompatibilidad con Hudi 0.13.1 y los tipos de marcas temporales. Es posible que se proporcione una solución en la próxima versión de Hudi.

Las tablas Hudi se clasifican de la siguiente manera, con implicaciones específicas para cada una de ellas:

- Copiar al escribir (CoW): los datos se almacenan en un formato de columnas (Parquet) y cada actualización crea una nueva versión de los archivos durante una escritura.

- Fusionar al leer (MoR): los datos se almacenan mediante la utilización de un formato que combina columnas (Parquet) y filas (Avro). Las actualizaciones se registran en archivos delta basados en filas y se compactan según sea necesario para crear nuevas versiones de los archivos en columnas.

Con los datasets de tipo CoW, cada vez que se produce una actualización de un registro, el archivo que contiene el registro se vuelve a escribir con los valores actualizados. Con un conjunto de datos de tipo MoR, cada vez que hay una actualización, Hudi escribe solo la fila correspondiente al registro modificado. MoR es más adecuado para cargas de trabajo con gran cantidad de escrituras o cambios y menos lecturas. CoW es más adecuado para cargas de trabajo con gran cantidad de lecturas con datos que cambian con menos frecuencia.

Hudi ofrece tres tipos de consulta para acceder a los datos:

- Consultas de instantáneas: consultas que ven la última instantánea de la tabla a partir de una acción de confirmación o compactación determinada. Para las tablas MoR, las consultas de instantáneas exponen el estado más reciente de la tabla mediante la combinación de los archivos base y delta del segmento de archivos más reciente en el momento de la consulta.
- Consultas progresivas: consultas que solo ven los nuevos datos escritos en la tabla, desde una confirmación o compactación determinada. Esto proporciona flujos de cambio de manera efectiva para habilitar canalizaciones de datos incrementales.
- Consultas optimizadas para lectura: para las tablas de MoR, las consultas ven compactados los datos más recientes. Para las tablas CoW, las consultas ven los últimos datos confirmados.

En el caso de las tablas Copiar al escribir, los rastreadores crean una sola tabla en el Data Catalog con el serde ReadOptimized `org.apache.hudi.hadoop.HoodieParquetInputFormat`.

En el caso de las tablas Fusionar al leer, el rastreador crea dos tablas en el Data Catalog para la misma ubicación de la tabla:

- Tabla con un sufijo `_ro` que utiliza el serde ReadOptimized `org.apache.hudi.hadoop.HoodieParquetInputFormat`.
- Una tabla con un sufijo `_rt` que utiliza el serde RealTime para realizar consultas instantáneas: `org.apache.hudi.hadoop.realtime.HoodieParquetRealtimeInputFormat`.

MongoDB y Amazon DocumentDB (compatible con MongoDB)

Las versiones 3.2 y posteriores de MongoDB son compatibles. Puede elegir rastrear sólo una pequeña muestra de los datos para reducir los tiempos de ejecución del rastreador.

Base de datos relacional

La autenticación se realiza con un nombre de usuario y una contraseña de base de datos. En función del tipo de motor de base de datos, puede elegir qué objetos se rastrean, como bases de datos, esquemas y tablas.

Snowflake

El rastreador de JDBC de Snowflake permite rastrear la tabla, la tabla externa, la vista y la vista materializada. La definición de vista materializada no se rellenará.

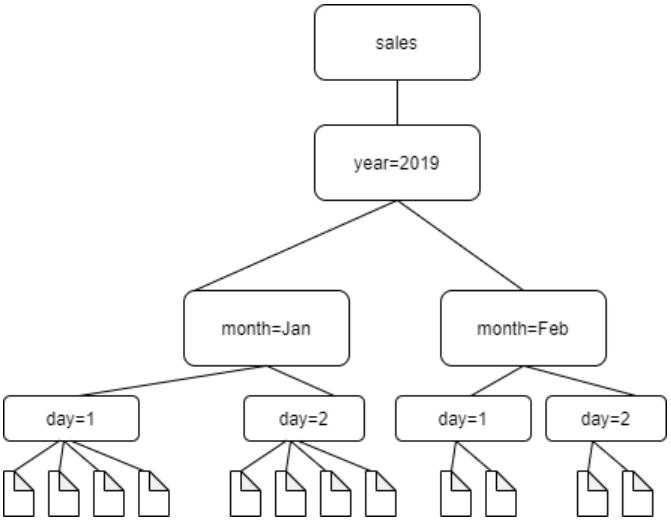
En el caso de las tablas externas de Snowflake, el rastreador solo llevará a cabo el rastreo si apunta a una ubicación de Amazon S3. Además del esquema de la tabla, el rastreador también rastreará la ubicación de Amazon S3, el formato de archivo y la salida como parámetros de tabla en la tabla del Catálogo de datos. Tenga en cuenta que la información de particiones de la tabla externa particionada no se rellena.

Actualmente, el proceso de ETL no es compatible con las tablas del Catálogo de datos creadas con el rastreador de Snowflake.

¿Cómo determina un rastreador cuándo crear particiones?

Cuando un rastreador de AWS Glue analiza Amazon S3 y detecta varias carpetas en un bucket, determina la raíz de una tabla en la estructura de carpetas y qué carpetas son particiones de una tabla. El nombre de la tabla se basa en el prefijo de Amazon S3 o el nombre de carpeta. Proporcione una ruta de inclusión que apunte al nivel de carpeta que se rastreará. Cuando la mayoría de los esquemas en el nivel de carpeta son similares, el rastreador crea particiones de una tabla en vez de tablas independientes. Para influir en el rastreador con el fin de que cree tablas independientes, agregue la carpeta raíz de cada tabla como un almacén de datos independiente al definir el rastreador.

Por ejemplo, considere la siguiente estructura de carpetas de Amazon S3.



Las rutas de acceso a las cuatro carpetas de nivel inferior son las siguientes:

```
S3://sales/year=2019/month=Jan/day=1
S3://sales/year=2019/month=Jan/day=2
S3://sales/year=2019/month=Feb/day=1
S3://sales/year=2019/month=Feb/day=2
```

Supongamos que el destino del rastreador está establecido en `Sales` y que todos los archivos en la carpeta `day=`*n* tienen el mismo formato (por ejemplo, JSON, no cifrado) y tienen los mismos esquemas o muy similares. El rastreador creará una sola tabla con cuatro particiones, con claves de partición `year`, `month` y `day`.

Por ejemplo, considere la siguiente estructura de Amazon S3:

```
s3://bucket01/folder1/table1/partition1/file.txt
s3://bucket01/folder1/table1/partition2/file.txt
s3://bucket01/folder1/table1/partition3/file.txt
s3://bucket01/folder1/table2/partition4/file.txt
s3://bucket01/folder1/table2/partition5/file.txt
```

Si los esquemas para los archivos en `table1` y `table2` son similares, y se define un almacén de datos individual en el rastreador con Include path (Ruta de inclusión) `s3://bucket01/folder1/`, el rastreador crea una sola tabla con dos columnas de claves de partición. La primera columna de clave de partición contiene `table1` y `table2`, y la segunda columna de clave de partición contiene `partition1` a `partition3` para la partición de la `table1`, y `partition4` y `partition5` para la partición de la `table2`. Para crear dos tablas independientes, defina el rastreador con dos

almacenes de datos. En este ejemplo, defina la primera ruta de inclusión como `s3://bucket01/folder1/table1/` y la segunda como `s3://bucket01/folder1/table2`.

> **Note**
>
> En Amazon Athena, cada tabla corresponde a un prefijo de Amazon S3 con todos los objetos que contiene. Si los objetos tienen diferentes esquemas, Athena no reconoce objetos distintos en el mismo prefijo como tablas independientes. Esto puede suceder si un rastreador crea varias tablas a partir del mismo prefijo de Amazon S3. Esto podría dar lugar a consultas en Athena que no devuelvan resultados. Para que Athena reconozca y consulte las tablas correctamente, cree el rastreador con una Include path (Ruta de inclusión)diferente para cada esquema de tabla en la estructura de carpetas de Amazon S3. Para obtener más información, consulte las Mejores prácticas de uso de Athena con AWS Glue y este artículo del Centro de conocimientos de AWS.

Requisitos previos del rastreador

El rastreador asume los permisos del rol de AWS Identity and Access Management (IAM) que se especifican al definirlo. Este rol de IAM debe tener permisos para extraer datos de su almacén de datos y escribir al Catálogo de datos. En la consola de AWS Glue solo se listan roles de IAM que tienen asociada una política de confianza para el servicio principal de AWS Glue. En la consola, también puede crear un rol de IAM con una política de IAM para obtener acceso a almacenes de datos de Amazon S3 a los que obtiene acceso el rastreador. Para obtener más información acerca de cómo proporcionar roles para AWS Glue, consulte Políticas basadas en la identidad para Glue AWS.

> **Note**
>
> Al rastrear un almacén de datos de Delta Lake, debe tener permisos de lectura y escritura en la ubicación de Simple Storage Service (Amazon S3).

Puede crear un rol para su rastreador y asociar las siguientes políticas:

- La política `AWSGlueServiceRole` administrada por AWS, que concede los permisos necesarios en el Catálogo de datos
- Política en línea que concede permisos en el origen de datos.

- Política en línea que concede permisos de `iam:PassRole` al rol.

Un enfoque más rápido es dejar que el asistente de rastreadores de la consola de AWS Glue cree un rol para usted. El rol que crea es específicamente para el rastreador e incluye la política `AWSGlueServiceRole` administrada por AWS, más la política en línea necesaria para el origen de datos especificado.

Si especifica un rol existente para un rastreador, asegúrese de que incluya la política `AWSGlueServiceRole` o equivalente (o una versión reducida de esta política), además de las políticas en línea requeridas. Por ejemplo, para un almacén de datos de Amazon S3, la política en línea sería, como mínimo, la siguiente:

```
{
   "Version": "2012-10-17",
    "Statement": [
        {
          "Effect": "Allow",
          "Action": [
              "s3:GetObject",
              "s3:PutObject"
          ],
          "Resource": [
              "arn:aws:s3:::bucket/object*"
          ]
        }
    ]
}
```

Para un almacén de datos de Amazon DynamoDB, la política sería, como mínimo, la siguiente:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:DescribeTable",
        "dynamodb:Scan"
      ],
      "Resource": [
        "arn:aws:dynamodb:region:account-id:table/table-name*"
```

```
      ]
    }
  ]
}
```

Si el rastreador lee datos de Amazon S3 cifrados por AWS Key Management Service (AWS KMS), entonces el rol de IAM debe tener permiso para descifrar la clave AWS KMS. Para obtener más información, consulte Paso 2: creación de un rol de IAM para AWS Glue.

Configuración de rastreadores

Un rastreador obtiene acceso al almacén de datos, extrae metadatos y crea definiciones de tablas en el AWS Glue Data Catalog. El panel Crawlers (Rastreadores) de la consola de AWS Glue lista todos los rastreadores que crea. La lista muestra el estado y las métricas desde la última ejecución de su rastreador.

> **Note**
>
> Si decide incorporar sus versiones de controladores JDBC, los rastreadores AWS Glue consumirán recursos en trabajos AWS Glue y en los buckets de Amazon S3 para garantizar que los controladores proporcionados se ejecuten en su entorno. El uso adicional de los recursos se reflejará en su cuenta. Además, proporcionar su propio controlador JDBC no significa que el rastreador pueda aprovechar todas las funciones del controlador. Los controladores están limitados a las propiedades descritas en Agregar una conexión AWS Glue.

Cómo configurar un rastreador

1. Inicie sesión en la AWS Management Console y abra la consola de AWS Glue en https://console.aws.amazon.com/glue/. Elija Crawlers (Rastreadores) en el panel de navegación.
2. Elija Crear rastreador y siga las instrucciones en el asistente Crear rastreador. El asistente le explicará los pasos necesarios para crear un rastreador. Si desea añadir clasificadores personalizados para definir el esquema, consulte Adición de clasificadores a un rastreador en AWS Glue.

Paso 1: configurar las propiedades del rastreador

Ingrese un nombre y una descripción para su rastreador (opcional). Si lo desea, puede etiquetar el rastreador con una clave de etiqueta y un valor de etiqueta opcional. Una vez creadas, las claves de etiqueta son de solo lectura. Utilice etiquetas en algunos recursos para que le resulte más fácil organizarlos e identificarlos. Para obtener más información, consulte las etiquetas de AWS en AWS Glue.

Nombre

El nombre puede contener letras (A-Z), números (0-9), guiones (-) o guiones bajos (_) y puede tener un máximo de 255 caracteres.

Descripción

La descripción puede tener una longitud máxima de 2048 caracteres.

Etiquetas

Utilice etiquetas para organizar e identificar los recursos. Para más información, consulte los siguientes temas:

- Etiquetas de AWS en AWS Glue

Paso 2: elegir orígenes de datos y clasificadores

Configuración del origen de datos

Seleccione la opción que corresponda para responder la pregunta ¿Sus datos ya están asignados a tablas de AWS Glue?: elija «Aún no» o «Sí». De manera predeterminada, está seleccionado 'Aún no'.

El rastreador puede acceder a los almacenes de datos directamente como origen de rastreo o puede utilizar tablas existentes del Catálogo de datos como origen. Si el rastreador utiliza tablas de catálogos existentes, este rastrea los almacenes de datos especificados por dichas tablas de catálogos.

- Not yet (Aún no): seleccione uno o varios orígenes de datos para rastrearlos. Un rastreador puede rastrear varios almacenes de datos de diferentes tipos (Amazon S3, JDBC, etc.).

  Solo puede configurar un almacén de datos por vez. Después de proporcionar la información de conexión e incluir rutas de acceso y patrones de exclusión, tendrá la opción de agregar otro almacén de datos.

- Yes (Sí): seleccione las tablas existentes de su Catálogo de datos de AWS Glue. Las tablas de catálogos especifican los almacenes de datos que se van a rastrear. El rastreador puede rastrear solo tablas de catálogos en una única ejecución; no puede combinar otros tipos de fuentes.

  Una razón habitual para especificar una tabla de catálogo como origen es que creó la tabla de forma manual (dado que ya conocía la estructura del almacén de datos) y quiere un rastreador para mantener la tabla actualizada, incluido el agregado de nuevas particiones. Para obtener información de otras razones, consulte Actualización de tablas del Catálogo de datos creadas de forma manual mediante rastreadores.

  Cuando especifique las tablas existentes como el tipo de origen de rastreador, se aplicarán las siguientes condiciones:

  - El nombre de la base de datos es opcional.
  - Solo se permiten las tablas de catálogos que especifiquen los almacenes de datos de Amazon S3 o Amazon DynamoDB.
  - No se crean nuevas tablas de catálogos cuando el rastreador se ejecuta. Las tablas existentes se actualizan según sea necesario, lo que incluye la adición de nuevas particiones.
  - Los objetos eliminados encontrados en los almacenes de datos se ignoran; no se eliminan tablas de catálogos. En su lugar, el rastreador escribe un mensaje de registro. (`SchemaChangePolicy.DeleteBehavior=LOG`)
  - La opción de la configuración del rastreador para crear un único esquema para cada ruta de Amazon S3 está activada de forma predeterminada y no se puede desactivar. (`TableGroupingPolicy=CombineCompatibleSchemas`) Para obtener más información, consulte Cómo crear un único esquema para cada ruta de inclusión de Amazon S3.
  - No se pueden combinar las tablas de catálogos como origen con cualquier otro tipo de origen (por ejemplo, Amazon S3 o Amazon DynamoDB).

Origen de datos

Seleccione o agregue la lista de orígenes de datos que el rastreador va a analizar.

(Opcional) Si elige JDBC como origen de datos, puede usar sus propios controladores JDBC al especificar el acceso a la conexión en el que se almacena la información del controlador.

Incluir ruta

Al evaluar lo que se va a incluir o excluir en un rastreo, un rastreador comienza por evaluar la ruta de inclusión necesaria. Para almacenes de datos relacionales, Amazon S3, MongoDB, MongoDB Atlas y Amazon DocumentDB (con compatibilidad con MongoDB), debe especificar una ruta de inclusión.

Para un almacén de datos de Amazon S3

Elija si desea especificar una ruta de esta cuenta o de una cuenta diferente y busque una ruta de Amazon S3.

Para los almacenes de datos de Amazon S3, la sintaxis de ruta de inclusión es `bucket-name/folder-name/file-name.ext`. Para rastrear todos los objetos de un bucket, debe especificar solo el nombre de bucket en la ruta de inclusión. EL patrón de exclusión es relativo a la ruta de inclusión

Para un almacén de datos de Delta Lake:

Especifique una o más rutas de Simple Storage Service (Amazon S3) a las tablas Delta como s3://*bucket*/*prefijo*/*objeto*.

Para un almacén de datos de Iceberg o Hudi

Especifique una o más rutas de Amazon S3 que contengan carpetas con metadatos de tablas de Iceberg o Hudi como s3://bucket/prefijo.

En el caso de un almacén de datos de Hudi, la carpeta de Hudi puede estar ubicada en una carpeta secundaria de la carpeta raíz. El rastreador escaneará todas las carpetas situadas debajo de una ruta para una carpeta Hudi.

En un almacén de datos JDBC

Ingrese *<database>*/*<schema>*/*<table>* o *<database>*/*<table>*, en función del producto de base de datos. Oracle Database y MySQL no permiten utilizar un esquema en la ruta. Puede sustituir *<esquema>* o *<tabla>* por el carácter de porcentaje (%). Por ejemplo, en una base de datos Oracle con el identificador del sistema (SID) `orcl`, escriba `orcl/%` para importar todas las tablas a las que el usuario especificado en la conexión tiene acceso.

> **Important**
>
> Este campo distingue entre mayúsculas y minúsculas.

Para un almacén de datos MongoDB, MongoDB Atlas o Amazon DocumentDB

Ingrese *database*/*collection*.

Para MongoDB, MongoDB Atlas y Amazon DocumentDB (con compatibilidad con MongoDB), la sintaxis es `database/collection`.

Para los almacenes de datos de JDBC, la sintaxis es `database-name/schema-name/table-name` o `database-name/table-name`. La sintaxis depende de si el motor de base de datos admite esquemas en una base de datos. Por ejemplo, en el caso de motores de base de datos como MySQL u Oracle, no especifique `schema-name` en la ruta de inclusión. Puede sustituir el signo de porcentaje (`%`) de un esquema o una tabla en la ruta de inclusión para representar todos los esquemas o todas las tablas de una base de datos. No se puede sustituir el signo de porcentaje (`%`) de base de datos en la ruta de inclusión.

Profundidad transversal máxima (solo para los almacenes de datos de Iceberg o Hudi)

Define la profundidad máxima de la ruta de Amazon S3 que el rastreador puede recorrer para descubrir la carpeta de metadatos de Iceberg o Hudi en la ruta de Amazon S3. El objetivo de este parámetro es limitar el tiempo de ejecución del rastreador. El valor predeterminado es 10 y el valor máximo es 20.

Patrones de exclusión

Estos le permiten excluir determinados archivos o tablas desde del rastreo. Una ruta de exclusión es relativa a la ruta de inclusión. Por ejemplo, para excluir una tabla en su almacén de datos de JDBC, escriba el nombre de la tabla en la ruta de exclusión.

Un rastreador se conecta a un almacén de datos de JDBC mediante una conexión de AWS Glue que contiene una cadena de conexión de URI de JDBC. El rastreador solo tiene acceso a los objetos en el motor de base de datos mediante el nombre de usuario y la contraseña de JDBC de la conexión de AWS Glue. El rastreador solo puede crear tablas a las que puede obtener acceso a través de la conexión de JDBC. Una vez que el rastreador obtiene acceso al motor de base de datos con el URI de JDBC, la ruta de inclusión se utiliza para determinar qué tablas de datos del motor de base de datos se crean en el Catálogo de datos. Por ejemplo, con MySQL, si especifica una ruta de inclusión de `MyDatabase/%`, todas las tablas en `MyDatabase` se crean en el Catálogo de datos. Al obtener acceso a Amazon Redshift, si especifica una ruta de inclusión de `MyDatabase/%`, todas las tablas en todos los esquemas de la base de datos `MyDatabase` se crean en el Catálogo de datos. Si especifica una ruta de inclusión de todas `MyDatabase/MySchema/%`, se crean todas las tablas de la base de datos `MyDatabase` y el esquema `MySchema`.

Después de especificar una ruta de inclusión, puede excluir objetos del rastreo que, de otro modo, su ruta de inclusión incluiría especificando uno o varios patrones de exclusión `glob` tipo Unix. Estos patrones se aplican a la ruta de inclusión para determinar qué objetos están excluidos. Estos patrones también se almacenan como una propiedad de las tablas creadas por el rastreador. AWS Glue Las extensiones PySpark, como `create_dynamic_frame.from_catalog`, leen las propiedades de la tabla y excluyen los objetos definidos por el patrón de exclusión.

AWS Glue admite los siguientes tipos de patrones `glob` en el patrón de exclusión.

Patrón de exclusión	Descripción
`*.csv`	Coincide con una ruta de Amazon S3 que representa un nombre de objeto de la carpeta actual que termina en `.csv`
`*.*`	Coincide con todos los nombres de objeto que contienen un punto
`*.{csv,avro}`	Coincide con los nombres de objeto que terminan con `.csv` o `.avro`
`foo.?`	Coincide con los nombres de objeto que comienzan por `foo.` a los que sigue una extensión de un solo carácter
`myfolder/*`	Coincide con los objetos en un nivel de subcarpeta desde `myfolder`, como `/myfolder/mysource`
`myfolder/*/*`	Coincide con los objetos en dos niveles de subcarpetas desde `myfolder`, como `/myfolder/mysource/data`
`myfolder/**`	Coincide con los objetos en todas las subcarpetas de `myfolder`, como `/myfolder/mysource/mydata` y `/myfolder/mysource/data`

Patrón de exclusión	Descripción
`myfolder**`	Coincide con la subcarpeta `myfolder` así como con los archivos debajo de `myfolder`, como `/myfolder` y `/myfolder/mydata.txt`
`Market*`	Coincide con las tablas en una base de datos de JDBC con nombres que comienzan por `Market`, como `Market_us` y `Market_fr`

AWS Glue interpreta los patrones de exclusión `glob` de la siguiente manera:

- El carácter de barra inclinada (`/`) es el delimitador para separar claves de Amazon S3 en una jerarquía de carpetas.
- El carácter asterisco (`*`) coincide con cero o varios caracteres de un componente de nombre sin superar límites de carpeta.
- Dos asteriscos (`**`) coinciden con cero o varios caracteres que superan límites de carpeta o esquema.
- El signo de interrogación (`?`) coincide exactamente con un carácter de un componente de nombre.
- El carácter de barra inversa (`\`) se usa para escapar caracteres que de otro modo se pueden interpretar como caracteres especiales. La expresión `\\` coincide con una sola barra inversa y `\{` coincide con una llave de apertura.
- Los corchetes `[ ]` crean una expresión de corchetes que coincide con un solo carácter de un componente de nombre fuera de un conjunto de caracteres. Por ejemplo, `[abc]` coincide con `a`, `b` o `c`. El guion (`-`) se puede usar para especificar un rango, por lo que `[a-z]` especifica un rango que coincide de la `a` a la `z` (inclusive). Estas formas se pueden mezclar, por lo que `[abce-g]` coincide con `a`, `b`, `c`, `e`, `f` o `g`. Si el carácter después del corchete (`[`) es un signo de exclamación (`!`), la expresión de corchetes se niega. Por ejemplo, `[!a-c]` coincide con cualquier carácter salvo con `a`, `b` o `c`.

  Dentro de una expresión de corchete, los caracteres `*`, `?` y `\` coinciden con ellos mismos. El guion (`-`) coincide consigo mismo si es el primer carácter dentro de los corchetes o si es el primer carácter después de `!` durante la negación.

- Las llaves (`{ }`) incluyen un grupo de subpatrones, donde el grupo coincide si cualquier subpatrón del grupo coincide. Una coma (`,`) se usa para separar los subpatrones. Los grupos no pueden estar anidados.
- Los puntos al inicio de los nombres de archivo se tratan como caracteres normales en las operaciones de coincidencia. Por ejemplo, el patrón de exclusión `*` coincide con el nombre de archivo `.hidden`.

Example Amazon S3 excluye patrones

Cada patrón de exclusión se evalúa con respecto a la ruta de inclusión. Por ejemplo, suponga que tiene la estructura de directorios de Amazon S3 siguiente:

```
/mybucket/myfolder/
   departments/
      finance.json
      market-us.json
      market-emea.json
      market-ap.json
   employees/
      hr.json
      john.csv
      jane.csv
      juan.txt
```

Dada la ruta de inclusión `s3://mybucket/myfolder/`, estos son algunos resultados de ejemplo para los patrones de exclusión:

Patrón de exclusión	Resultados
`departments/**`	Excluye todas las carpetas y archivos situados debajo de `departments` e incluye la carpeta `employees` y sus archivos
`departments/market*`	Excluye `market-us.json` , `market-emea.json` y `market-ap.json`
`**.csv`	Excluye todos los objetos situados debajo de `myfolder` con un nombre terminado con `.csv`

Patrón de exclusión	Resultados
`employees/*.csv`	Excluye todos los archivos `.csv` de la carpeta `employees`

Example Excluir un subconjunto de particiones de Amazon S3

Supongamos que sus datos se particionan por día de modo que cada uno de los días de un año esté en una partición de Amazon S3 independiente. En enero de 2015, hay 31 particiones. Ahora, para rastrear datos durante solo la primera semana de enero, debe excluir todas las particiones excepto del día 1 al 7:

```
2015/01/{[!0],0[8-9]}**, 2015/0[2-9]/**, 2015/1[0-2]/**
```

Veamos las partes de este patrón de glob. La primera parte, `2015/01/{[!0],0[8-9]}**`, excluye todos los días que no comienzan por "0", además de los días 08 y 09 del mes 01 del año 2015. Tenga en cuenta que "**" se usa como sufijo del patrón de número de día y supera límites de carpeta hasta alcanzar carpetas de nivel más bajo. Si "*" se usa, los niveles de carpeta más bajos no se excluyen.

La segunda parte, `2015/0[2-9]/**`, excluye días de los meses comprendidos entre el 02 y el 09 del año 2015.

La tercera parte, `2015/1[0-2]/**`, excluye días de los meses 10, 11 y 12 del año 2015.

Example Patrones de exclusión de JDBC

Supongamos que rastrea una base de datos de JDBC con la siguiente estructura de esquema:

```
MyDatabase/MySchema/
   HR_us
   HR_fr
   Employees_Table
   Finance
   Market_US_Table
   Market_EMEA_Table
   Market_AP_Table
```

Dada la ruta de inclusión `MyDatabase/MySchema/%`, estos son algunos resultados de ejemplo para los patrones de exclusión:

Patrón de exclusión	Resultados
`HR*`	Excluye las tablas con nombres que comienzan por `HR`
`Market_*`	Excluye las tablas con nombres que comienzan por `Market_`
`**_Table`	Excluye todas las tablas con nombres que finalizan con `_Table`

Parámetros adicionales de origen de rastreadores

Cada tipo de origen requiere un conjunto diferente de parámetros adicionales. La siguiente lista no es exhaustiva:

Connection

Seleccione o agregue una conexión de AWS Glue. Para obtener más información acerca de las conexiones, consulte Conexión a datos.

Metadatos adicionales: opcionales (para almacenes de datos de JDBC)

Seleccione propiedades de metadatos adicionales para que el rastreador las rastree.

- Comments (Comentarios): se rastrean los comentarios asociados de tabla y columna.
- Raw types (Tipos sin procesar): se mantienen los tipos de datos sin procesar de las columnas de la tabla en metadatos adicionales. Como comportamiento predeterminado, el rastreador traduce los tipos de datos sin procesar a tipos compatibles con Hive.

Nombre de clase de controlador JDBC: opcional (para almacenes de datos JDBC)

Escriba un nombre de clase de controlador JDBC personalizado para que el rastreador se conecte al origen de datos:

- Postgres: org.postgresql.Driver
- MySQL: com.mysql.jdbc.Driver, com.mysql.cj.jdbc.Driver
- Redshift: com.amazon.redshift.jdbc.Driver, com.amazon.redshift.jdbc42.Driver

- Oracle: oracle.jdbc.driver.OracleDriver
- SQL Server: com.microsoft.sqlserver.jdbc.SQLServerDriver

Ruta S3 del controlador JDBC: opcional (para almacenes de datos JDBC)

Elija una ruta de Amazon S3 existente para un archivo `.jar`. Aquí es donde se almacenará el archivo `.jar` cuando se utilice un controlador JDBC personalizado para que el rastreador se conecte al origen de datos.

Habilitación del muestreo de datos (solo en almacenes de datos Amazon DynamoDB, MongoDB, MongoDB Atlas y Amazon DocumentDB)

Seleccione si desea rastrear solo una muestra de datos. Si no lo selecciona, se rastreará toda la tabla. Escanear todos los registros puede tardar mucho tiempo cuando la tabla no es una tabla de alto rendimiento.

Crear tablas para consultar (solo para almacenes de datos de Delta Lake)

Seleccione cómo desea crear las tablas de Delta Lake:

- Crear tablas nativas: se permite la integración con los motores de consulta que permiten consultar directamente el registro de transacciones de Delta.
- Create Symlink tables (Crear tablas de enlaces simbólicos): se crea una carpeta de manifiesto de enlaces simbólicos con los archivos de manifiesto particionados mediante las claves de partición en función de los parámetros de configuración especificados.

Scanning rate (Velocidad de análisis): opcional (solo en almacenes de datos de DynamoDB)

Especifique el porcentaje de las unidades de capacidad de lectura de tablas de DynamoDB que utilizará el rastreador. Unidades de capacidad de lectura es un término definido por DynamoDB y es un valor numérico que actúa como limitador de velocidad del número de lecturas que se pueden realizar en esa tabla por segundo. Introduzca un valor comprendido entre 0,1 y 1,5. Si no se especifica, el valor predeterminado es de 0,5% para las tablas aprovisionadas y 1/4 de la capacidad máxima configurada para las tablas bajo demanda. Tenga en cuenta que solo se debe utilizar el modo de capacidad aprovisionada con rastreadores de AWS Glue.

> **Note**
>
> Para los almacenes de datos de DynamoDB, establezca el modo de capacidad aprovisionada para procesar las lecturas y escrituras en las tablas. El rastreador de AWS Glue no debe utilizarse con el modo de capacidad bajo demanda.

Network connection (Conexión de red): opcional (solo para almacenes de datos de Amazon S3)

Si lo desea, incluya una conexión de red para utilizarla con este destino de Amazon S3. Tenga en cuenta que cada rastreador se limita a una conexión de red, por lo que cualquier otro destino de Amazon S3 también utilizará la misma conexión (o ninguna, si se deja en blanco).

Para obtener más información acerca de las conexiones, consulte Conexión a datos.

Muestrear solo un subconjunto de archivos y tamaño de la muestra (solo para almacenes de datos de Amazon S3)

Establece el número de archivos de cada carpeta que se van a rastrear al rastrear archivos de ejemplo en un conjunto de datos. Cuando esta característica está activada, en lugar de rastrear todos los archivos de este conjunto de datos, el rastreador selecciona aleatoriamente algunos archivos en cada carpeta para rastrear.

El rastreador de muestreo es el más adecuado para clientes que tienen conocimientos previos sobre sus formatos de datos y saben que los esquemas de sus carpetas no cambian. Activar esta característica reducirá significativamente el tiempo de ejecución del rastreador.

Un valor válido es un entero entre 1 y 249. Si no se especifica, se rastrean todos los archivos.

Ejecuciones posteriores del rastreador

Este campo es un campo global que afecta a todos los orígenes de datos de Amazon S3.

- Crawl all sub-folders (Rastrear todas las subcarpetas): se vuelven a rastrear todas las carpetas con cada rastreo posterior.
- Crawl new sub-folders only (Rastrear solo las subcarpetas nuevas): solo se rastrearán las carpetas de Amazon S3 que se hayan agregado desde el último rastreo. Si los esquemas son compatibles, se agregarán nuevas particiones a las tablas existentes. Para obtener más información, consulte the section called “Rastreos progresivos para agregar nuevas particiones”.
- Crawl based on events (Rastreo basado en eventos): confíe en los eventos de Amazon S3 para controlar qué carpetas rastrear. Para obtener más información, consulte the section called “Aceleración de los rastreadores mediante las notificaciones de eventos de Amazon S3”.

Custom classifiers (Clasificadores personalizados): opcionales

Defina los clasificadores personalizados antes de definir los rastreadores. Un clasificador comprueba si un determinado archivo está en un formato que puede el rastreador administrar. En caso afirmativo, el clasificador crea un esquema en forma de un objeto `StructType` que coincida con formato de datos.

Para obtener más información, consulte Adición de clasificadores a un rastreador en AWS Glue.

Paso 3: Establecer configuración de seguridad

Rol de IAM

El rastreador asume este rol. Debe tener permisos a la política `AWSGlueServiceRole` administrada por AWS. En el caso de los orígenes de Amazon S3 y DynamoDB, también debe tener permisos para acceder al almacén de datos. Si el rastreador lee datos de Amazon S3 cifrados por AWS Key Management Service (AWS KMS), entonces el rol debe tener permiso para descifrar la clave AWS KMS.

Para un almacén de datos de Amazon S3, los permisos adicionales asociados al rol serían similares a los siguientes:

```
{
   "Version": "2012-10-17",
    "Statement": [
        {
          "Effect": "Allow",
          "Action": [
              "s3:GetObject",
              "s3:PutObject"
          ],
          "Resource": [
              "arn:aws:s3:::bucket/object*"
          ]
        }
    ]
}
```

Para un almacén de datos de Amazon DynamoDB, los permisos adicionales asociados al rol serían similares a los siguientes:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:DescribeTable",
```

```
        "dynamodb:Scan"
      ],
      "Resource": [
        "arn:aws:dynamodb:region:account-id:table/table-name*"
      ]
    }
  ]
}
```

Para agregar su propio controlador JDBC, es necesario agregar permisos adicionales.

- Conceda permisos para las siguientes acciones de trabajos: `CreateJob`, `DeleteJob`, `GetJob`, `GetJobRun`, `StartJobRun`.
- Conceda permisos para todas las acciones de Amazon S3: `s3:DeleteObjects`, `s3:GetObject`, `s3:ListBucket`, `s3:PutObject`.

> **Note**
>
> `s3:ListBucket` no es necesaria si la política de bucket de Amazon S3 está deshabilitada.

- Conceda el acceso principal del servicio al bucket o la carpeta en la política de Amazon S3.

Ejemplo de política de Amazon S3:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::bucket-name/driver-parent-folder/driver.jar",
                "arn:aws:s3:::bucket-name"
            ]
        }
```

```
    ]
}
```

AWS Glue crea las siguientes carpetas (`_crawler` y `_glue_job_crawler` al mismo nivel que el controlador JDBC en su bucket de Amazon S3). Por ejemplo, si la ruta del controlador es `<s3-path/driver_folder/driver.jar>`, entonces se crearán las siguientes carpetas si aún no existen:

- <s3-path/driver_folder/_crawler>
- <s3-path/driver_folder/_glue_job_crawler>

De forma opcional, puede agregar una configuración de seguridad a un rastreador para especificar opciones de cifrado en reposo.

Para obtener más información, consulte Paso 2: creación de un rol de IAM para AWS Glue y Gestión de identidad y acceso para AWS Glue.

Lake Formation configuration (Configuración de Lake Formation): opcional

Permita que el rastreador utilice credenciales de Lake Formation para rastrear el origen de datos.

Al marcar Use Lake Formation credentials for crawling S3 data source (Utilizar credenciales de Lake Formation para rastrear un origen de datos S3), el rastreador podrá usar las credenciales de Lake Formation para rastrear el origen de datos. Si el origen de datos pertenece a otra cuenta, debe proporcionar el ID de la cuenta registrada. De lo contrario, el rastreador solo rastreará los orígenes de datos asociados a la cuenta. Solo se aplica a los orígenes de datos del Catálogo de datos y Amazon S3.

Security configuration (Configuración de seguridad): opcional

La configuración incluye ajustes de seguridad. Para más información, consulte los siguientes temas:

- Cifrado de datos escritos por AWS Glue

> **Note**
>
> Una vez que se establezca una configuración de seguridad en un rastreador, puede modificarla, pero no eliminarla. Para reducir el nivel de seguridad de una rastreador,

establezca de manera explícita la característica de seguridad en `DISABLED` dentro de la configuración o cree un nuevo rastreador.

Paso 4: establecer salida y programación

Configuración de la salida

Las opciones incluyen la forma en la que el rastreador debe gestionar los cambios detectados en un esquema, los objetos eliminados en el almacén de datos y mucho más. Para obtener más información, consulte Personalización del comportamiento del rastreador

Propiedad Schedule de rastreador

Puede ejecutar un rastreador bajo demanda o definir programaciones basadas en tiempo para los rastreadores y los trabajos de AWS Glue. La definición de estas programaciones utiliza sintaxis cron del tipo Unix. Para obtener más información, consulte Programación de un rastreador de AWS Glue.

Paso 5: Revisar y crear

Revise la configuración del rastreador que estableció y cree el rastreador.

Adición de clasificadores a un rastreador en AWS Glue

Un clasificador lee los datos en un almacén de datos. Si reconoce el formato de los datos, genera un esquema. El clasificador también devuelve un número de certeza para indicar el grado de certeza del reconocimiento del formato.

AWS Glue proporciona un conjunto de clasificadores integrados, pero también puede crear clasificadores personalizados. AWS Glue invoca clasificadores personalizados en primer lugar, en el orden especificado en su definición del rastreador. Según los resultados devueltos a partir de los clasificadores personalizados, AWS Glue también podría invocar clasificadores integrados. Si un clasificador devuelve `certainty=1.0` durante el procesamiento, indica que puede crear el esquema correcto con una seguridad del 100 por ciento. A continuación, AWS Glue usa la salida de ese clasificador.

Si ningún clasificador devuelve `certainty=1.0`, AWS Glue usará la salida del clasificador con la mayor certeza. Si ningún clasificador devuelve una certeza mayor que `0.0`, AWS Glue devolverá la cadena de clasificación predeterminada de `UNKNOWN`.

¿Cuándo uso un clasificador?

Se usan clasificadores al rastrear un almacén de datos para definir tablas de metadatos en el AWS Glue Data Catalog. Puede configurar su rastreador con un conjunto ordenado de clasificadores. Cuando el rastreador invoca un clasificador, el clasificador determina si se reconocen los datos. Si el clasificador no puede reconocer los datos o no es 100 % seguro, el rastreador invoca el siguiente clasificador de la lista para determinar si puede reconocer los datos.

Para obtener información adicional acerca de cómo crear un clasificador mediante la consola de AWS Glue, consulte Trabajo con clasificadores en la consola de AWS Glue.

Clasificadores personalizados

La salida de un clasificador incluye una cadena que indica la clasificación o el formato del archivo (por ejemplo, `json`) y el esquema del archivo. En el caso de los clasificadores personalizados, defina la lógica para crear el esquema en función del tipo de clasificador. Los tipos de clasificador incluyen la definición de esquemas en función de patrones de grok, etiquetas XML y rutas JSON.

Si cambia una definición de clasificador, no se vuelven a clasificar los datos que se rastrearon anteriormente con el clasificador. Un rastreador realiza un seguimiento de datos rastreados anteriormente. Los datos nuevos se clasifican con el clasificador actualizado, lo que podría dar lugar a un esquema actualizado. Si el esquema de sus datos ha evolucionado, actualice el clasificador para tener en cuenta los cambios de esquema cuando se ejecute el rastreador. Para volver a clasificar los datos con el fin de corregir un clasificador incorrecto, cree un nuevo rastreador con el clasificador actualizado.

Para obtener más información acerca de la creación de clasificadores personalizados en AWS Glue, consulte Escritura de clasificadores personalizados.

> **Note**
>
> Si uno de los clasificadores integrados reconoce el formato de sus datos, no es necesario que cree un clasificador personalizado.

Clasificadores integrados en AWS Glue

AWS Glue proporciona clasificadores integrados para diversos formatos, incluidos JSON, CSV, registros web y muchos sistemas de bases de datos.

Si AWS Glue no encuentra un clasificador personalizado que se adapte mejor al formato de datos de entrada con una seguridad del 100 por ciento, invoca los clasificadores integrados en el orden mostrado en la siguiente tabla. Los clasificadores integrados devuelven un resultado para indicar si el formato coincide (`certainty=1.0`) o no coincide (`certainty=0.0`). El primer clasificador con `certainty=1.0` proporciona la cadena de clasificación y el esquema para una tabla de metadatos en su Data Catalog.

Tipo de clasificador	Cadena de clasificación	Notas
Apache Avro	`avro`	Lee el esquema al principio del archivo para determinar el formato.
Apache ORC	`orc`	Lee los metadatos de archivo para determinar el formato.
Apache Parquet	`parquet`	Lee el esquema al final del archivo para determinar el formato.
JSON	`json`	Lee el principio del archivo para determinar el formato.
JSON binario	`bson`	Lee el principio del archivo para determinar el formato.
XML	`xml`	Lee el principio del archivo para determinar el formato. AWS Glue determina el esquema de la tabla basado en etiquetas XML del documento. Para obtener información acerca de la creación de un clasificador XML personalizado para especificar filas en el documento, consulte Escritura de clasificadores personalizados XML.
Amazon Ion	`ion`	Lee el principio del archivo para determinar el formato.
Registro Apache combinado	`combined_apache`	Determina los formatos de log a través de un patrón de grok.

Tipo de clasificador	Cadena de clasificación	Notas
Registro Apache	`apache`	Determina los formatos de log a través de un patrón de grok.
Registro de kernel de Linux	`linux_kernel`	Determina los formatos de log a través de un patrón de grok.
Registro de Microsoft	`microsoft_log`	Determina los formatos de log a través de un patrón de grok.
Registro de Ruby	`ruby_logger`	Lee el principio del archivo para determinar el formato.
Registro de Squid 3.x	`squid`	Lee el principio del archivo para determinar el formato.
Registro de monitorización de Redis	`redismonlog`	Lee el principio del archivo para determinar el formato.
Registro de Redis	`redislog`	Lee el principio del archivo para determinar el formato.
CSV	`csv`	Comprueba los siguientes delimitadores: coma (,), barra vertical (|), tabulación (\t), punto y coma (;) y Ctrl-A (\u0001). Ctrl-A es el carácter de control Unicode para `Start Of Heading`.
Amazon Redshift	`redshift`	Usa la conexión de JDBC para importar metadatos.
MySQL	`mysql`	Usa la conexión de JDBC para importar metadatos.
PostgreSQL	`postgresql`	Usa la conexión de JDBC para importar metadatos.
Base de datos de Oracle	`oracle`	Usa la conexión de JDBC para importar metadatos.

Tipo de clasificador	Cadena de clasificación	Notas
Microsoft SQL Server	`sqlserver`	Usa la conexión de JDBC para importar metadatos.
Amazon DynamoDB	`dynamodb`	Lee datos de la tabla de DynamoDB.

Los archivos en los siguientes formatos comprimidos se pueden clasificar:

- ZIP (compatible con archivos que solo contengan un único archivo). Tenga en cuenta que Zip no es totalmente compatible en otros servicios (por el archivo).
- BZIP
- GZIP
- LZ4
- Snappy (compatible con los formatos Snappy estándar y nativo de Hadoop)

Clasificador de CSV integrado

El clasificador de CSV integrado analiza el contenido del archivo CSV para determinar el esquema de una tabla de AWS Glue. Este clasificador comprueba los siguientes delimitadores:

- Coma (,)
- Barra vertical (|)
- Tabulador (\t)
- Punto y coma (;)
- Ctrl-A (\u0001)

  Ctrl-A es el carácter de control Unicode para `Start Of Heading`.

Para clasificarse como CSV, el esquema de tabla debe tener al menos dos columnas y dos filas de datos. El clasificador de CSV utiliza una serie de funciones heurísticas para determinar si un encabezado está en un determinado archivo. Si el clasificador no puede determinar un encabezado a partir de la primera fila de datos, los encabezados de columna se muestran como `col1`, `col2`, `col3`,

etc. El clasificador de CSV determina si debe deducir un encabezado mediante la evaluación de las siguientes características del archivo:

- Cada columna en un posible encabezado se analiza como un tipo de datos STRING.
- Excepto en la última, cada columna en un posible encabezado incluye contenido de menos de 150 caracteres. Para permitir un delimitador final, la última columna puede estar vacía en el archivo.
- Cada columna de un encabezado potencial debe cumplir los requisitos `regex` de AWS Glue correspondientes a un nombre de columna.
- La fila de encabezado debe estar suficientemente diferenciada de las filas de datos. Para determinarlo, una o varias de las filas deben analizarse como un tipo distinto de STRING. Si todas las columnas son de tipo STRING, la primera fila de datos que no esté suficientemente diferenciada de las filas posteriores se usará como el encabezado.

> **Note**
>
> Si el clasificador de CSV integrado no crea la tabla de AWS Glue como desea, es posible que deba usar una de las siguientes alternativas:
>
> - Cambie los nombres de columnas en el Data Catalog, establezca `SchemaChangePolicy` en LOG y defina la configuración de salida de partición en `InheritFromTable` para las futuras ejecuciones de rastreador.
> - Cree un clasificador de grok para analizar los datos y asignar las columnas que desee.
> - El clasificador de CSV crea tablas haciendo referencia a `LazySimpleSerDe` como la biblioteca de serialización, lo que es una buena opción para la inferencia de tipo. Sin embargo, si los datos CSV contienen cadenas entre comillas, edite la definición de tabla y cambie la biblioteca SerDe a `OpenCSVSerDe`. Ajuste los tipos inferidos a STRING, establezca `SchemaChangePolicy` en LOG y defina la configuración de salida de partición en `InheritFromTable` para las futuras ejecuciones de rastreador. Para obtener más información acerca de las bibliotecas SerDe, consulte Referencia de SerDe en la Guía del usuario de Amazon Athena.

Escritura de clasificadores personalizados

Puede proporcionar un clasificador personalizado para clasificar los datos de AWS Glue. Puede crear un clasificador personalizado con un patrón grok, con una etiqueta XML, con la notación de objetos

JavaScript (JSON) o con valores separados por comas (CSV). Un rastreador de AWS Glue llama a un clasificador personalizado. Si el clasificador reconoce los datos, devuelve la clasificación y el esquema de los datos al rastreador. Puede que tenga que definir un clasificador personalizado si sus datos no coinciden con ningún clasificador integrado o si desea personalizar las tablas creadas por el rastreador.

Para obtener información adicional acerca de cómo crear un clasificador mediante la consola de AWS Glue, consulte Trabajo con clasificadores en la consola de AWS Glue.

AWS Glue ejecuta los clasificadores personalizados antes que los integrados, en el orden especificado. Cuando un rastreador encuentra un clasificador que coincide con los datos, la cadena de clasificación y el esquema se usan en la definición de las tablas escritas en su AWS Glue Data Catalog.

Temas

- Escritura de clasificadores personalizados de Grok
- Escritura de clasificadores personalizados XML
- Escritura de clasificadores personalizados JSON
- Escritura de clasificadores personalizados CSV

Escritura de clasificadores personalizados de Grok

Grok es una herramienta que se usa para analizar datos textuales con un patrón coincidente. Un patrón de grok es un conjunto designado de expresiones regulares (regex) que se usan para que los datos coincidan con una línea a la vez. AWS Glue usa patrones de grok para inferir el esquema de sus datos. Cuando un patrón de grok coincide con sus datos, AWS Glue usa el patrón para determinar la estructura de sus datos y mapearlos en campos.

AWS Glue proporciona muchos patrones integrados o usted puede definir los suyos propios. Puede crear un patrón de grok mediante patrones integrados y personalizados en su definición personalizada del clasificador. Puede adaptar un patrón de grok para clasificar formatos de archivos de texto personalizados.

> **Note**
>
> Los clasificadores personalizados de grok de AWS Glue usan la biblioteca de serialización `GrokSerDe` para las tablas creadas en AWS Glue Data Catalog. Si usa AWS Glue Data

Catalog con Amazon Athena, Amazon EMR o Redshift Spectrum, consulte la documentación acerca de esos servicios para obtener información acerca del soporte de `GrokSerDe`. Actualmente, podría tener problemas al consultar las tablas creadas con `GrokSerDe` desde Amazon EMR y Redshift Spectrum.

Esta es la sintaxis básica para los componentes de un patrón de grok:

```
%{PATTERN:field-name}
```

Los datos que coinciden con el elemento `PATTERN` designado se mapean a la columna `field-name` del esquema, con un tipo de datos predeterminado de `string`. Si lo prefiere, el tipo de datos para el campo se puede convertir en `byte`, `boolean`, `double`, `short`, `int`, `long` o `float` en el esquema resultante.

```
%{PATTERN:field-name:data-type}
```

Por ejemplo, para convertir un campo `num` en un tipo de datos `int`, puede usar este patrón:

```
%{NUMBER:num:int}
```

Los patrones se pueden componer de otros patrones. Por ejemplo, puede tener un patrón para una marca temporal `SYSLOG` definida utilizando patrones para el mes, el día del mes y la hora (por ejemplo, `Feb 1 06:25:43`). Para estos datos, puede definir el siguiente patrón:

```
SYSLOGTIMESTAMP %{MONTH} +%{MONTHDAY} %{TIME}
```

Note

Los patrones de grok pueden procesar una única línea a la vez. No se admiten los patrones de varias líneas. Asimismo, no se admiten los saltos de línea dentro de un patrón.

Valores personalizados del clasificador en AWS Glue

Al definir un clasificador de grok, proporciona los siguientes valores a AWS Glue para crear el clasificador personalizado.

Nombre

Nombre del clasificador.

Clasificación

La cadena de texto que se escribe para describir el formato de los datos que se clasifican; por ejemplo, `special-logs`.

Patrón de Grok

El conjunto de patrones que se aplican al almacén de datos para determinar si existe una coincidencia. Estos patrones proceden de AWS Gluepatrones integrados de y cualquier patrón personalizado que defina.

A continuación se muestra un ejemplo de un patrón de grok:

```
%{TIMESTAMP_ISO8601:timestamp} \[%{MESSAGEPREFIX:message_prefix}\]
 %{CRAWLERLOGLEVEL:loglevel} : %{GREEDYDATA:message}
```

Cuando los datos coinciden con `TIMESTAMP_ISO8601`, se crea una columna de esquema `timestamp`. El comportamiento es similar para los otros patrones designados del ejemplo.

Patrones personalizados

Patrones personalizados opcionales definidos por usted. El patrón de grok que clasifica sus datos hace referencia a estos patrones. Puede hacer referencia a estos patrones personalizados en el patrón de grok aplicado a sus datos. Cada patrón de componente personalizado debe estar en una línea independiente. La sintaxis Expresión regular (regex) se usa para definir el patrón.

A continuación se muestra un ejemplo del uso de patrones personalizados:

```
CRAWLERLOGLEVEL (BENCHMARK|ERROR|WARN|INFO|TRACE)
MESSAGEPREFIX .*-.*-.*-.*-.*
```

El primer patrón designado personalizado, `CRAWLERLOGLEVEL`, es una coincidencia si los datos coinciden con una de las cadenas enumeradas. El segundo patrón personalizado, `MESSAGEPREFIX`, intenta coincidir con una cadena prefijo del mensaje.

AWS Glue realiza un seguimiento de la hora de creación, la hora de la última actualización y la versión de su clasificador.

Patrones integrados de AWS Glue

AWS Glue proporciona muchos patrones comunes que puede usar para crear un clasificador personalizado. Puede añadir un patrón designado a `grok pattern` en una definición del clasificador.

La siguiente lista consta de una línea para cada patrón. En cada línea, al nombre del patrón le sigue su definición. Para definir el patrón, se utiliza la sintaxis de las expresiones regulares (regex).

```
#<noloc>&GLU;</noloc> Built-in patterns
 USERNAME [a-zA-Z0-9._-]+
 USER %{USERNAME:UNWANTED}
 INT (?:[+-]?(?:[0-9]+))
 BASE10NUM (?<![0-9.+-])(?>[+-]?(?:(?:[0-9]+(?:\.[0-9]+)?)|(?:\.[0-9]+)))
 NUMBER (?:%{BASE10NUM:UNWANTED})
 BASE16NUM (?<![0-9A-Fa-f])(?:[+-]?(?:0x)?(?:[0-9A-Fa-f]+))
 BASE16FLOAT \b(?<![0-9A-Fa-f.])(?:[+-]?(?:0x)?(?:(?:[0-9A-Fa-f]+(?:\.[0-9A-Fa-f]*)?)|
(?:\.[0-9A-Fa-f]+)))\b
 BOOLEAN (?i)(true|false)

 POSINT \b(?:[1-9][0-9]*)\b
 NONNEGINT \b(?:[0-9]+)\b
 WORD \b\w+\b
 NOTSPACE \S+
 SPACE \s*
 DATA .*?
 GREEDYDATA .*
 #QUOTEDSTRING (?:(?<!\\)(?:"(?:\\.|[^\\"])*"|(?:'(?:\\.|[^\\'])*')|(?:`(?:\\.|[^\
\`])*`)))
 QUOTEDSTRING (?>(?<!\\)(?>"(?>\\.|[^\\"]+)+"|""|(?>'(?>\\.|[^\\']+)+')|''|(?>`(?>\\.|
[^\\`]+)+`)|``))
 UUID [A-Fa-f0-9]{8}-(?:[A-Fa-f0-9]{4}-){3}[A-Fa-f0-9]{12}

 # Networking
 MAC (?:%{CISCOMAC:UNWANTED}|%{WINDOWSMAC:UNWANTED}|%{COMMONMAC:UNWANTED})
 CISCOMAC (?:(?:[A-Fa-f0-9]{4}\.){2}[A-Fa-f0-9]{4})
 WINDOWSMAC (?:(?:[A-Fa-f0-9]{2}-){5}[A-Fa-f0-9]{2})
 COMMONMAC (?:(?:[A-Fa-f0-9]{2}:){5}[A-Fa-f0-9]{2})
 IPV6 ((([0-9A-Fa-f]{1,4}:){7}([0-9A-Fa-f]{1,4}|:))|(([0-9A-Fa-f]{1,4}:){6}(:[0-9A-
Fa-f]{1,4}|((25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)(\.(25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d))
{3})|:))|(([0-9A-Fa-f]{1,4}:){5}(((:[0-9A-Fa-f]{1,4}){1,2})|:((25[0-5]|2[0-4]\d|1\d
\d|[1-9]?\d)(\.(25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)){3})|:))|(([0-9A-Fa-f]{1,4}:){4}(((:
[0-9A-Fa-f]{1,4}){1,3})|((:[0-9A-Fa-f]{1,4})?:((25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)(\.
```

```
(25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)){3}))|:))|(([0-9A-Fa-f]{1,4}:){3}(((:[0-9A-Fa-f]
{1,4}){1,4})|((:[0-9A-Fa-f]{1,4}){0,2}:((25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)(\.(25[0-5]|
2[0-4]\d|1\d\d|[1-9]?\d)){3}))|:))|(([0-9A-Fa-f]{1,4}:){2}(((:[0-9A-Fa-f]{1,4}){1,5})|
((:[0-9A-Fa-f]{1,4}){0,3}:((25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)(\.(25[0-5]|2[0-4]\d|1\d
\d|[1-9]?\d)){3}))|:))|(([0-9A-Fa-f]{1,4}:){1}(((:[0-9A-Fa-f]{1,4}){1,6})|((:[0-9A-Fa-
f]{1,4}){0,4}:((25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)(\.(25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d))
{3}))|:))|(:(((:[0-9A-Fa-f]{1,4}){1,7})|((:[0-9A-Fa-f]{1,4}){0,5}:((25[0-5]|2[0-4]\d|
1\d\d|[1-9]?\d)(\.(25[0-5]|2[0-4]\d|1\d\d|[1-9]?\d)){3}))|:)))(%.+)?
 IPV4 (?<![0-9])(?:(?:25[0-5]|2[0-4][0-9]|[0-1]?[0-9]{1,2})[.](?:25[0-5]|2[0-4][0-9]|
[0-1]?[0-9]{1,2})[.](?:25[0-5]|2[0-4][0-9]|[0-1]?[0-9]{1,2})[.](?:25[0-5]|2[0-4][0-9]|
[0-1]?[0-9]{1,2}))(?![0-9])
 IP (?:%{IPV6:UNWANTED}|%{IPV4:UNWANTED})
 HOSTNAME \b(?:[0-9A-Za-z][0-9A-Za-z-_]{0,62})(?:\.(?:[0-9A-Za-z][0-9A-Za-z-_]
{0,62}))*(\.?|\b)
 HOST %{HOSTNAME:UNWANTED}
 IPORHOST (?:%{HOSTNAME:UNWANTED}|%{IP:UNWANTED})
 HOSTPORT (?:%{IPORHOST}:%{POSINT:PORT})

 # paths
 PATH (?:%{UNIXPATH}|%{WINPATH})
 UNIXPATH (?>/(?>[\w_%!$@:.,~-]+|\\.)*)+
 #UNIXPATH (?<![\w\/])(?:/[^\/\s?*]*)+
 TTY (?:/dev/(pts|tty([pq])?)(\w+)?/?(?:[0-9]+))
 WINPATH (?>[A-Za-z]+:|\\)(?:\\[^\\?*]*)+
 URIPROTO [A-Za-z]+(\+[A-Za-z+]+)?
 URIHOST %{IPORHOST}(?::%{POSINT:port})?
 # uripath comes loosely from RFC1738, but mostly from what Firefox
 # doesn't turn into %XX
 URIPATH (?:/[A-Za-z0-9$.+!*'(){},~:;=@#%_\-]*)+
 #URIPARAM \?(?:[A-Za-z0-9]+(?:=(?:[^&]*))?(?:&(?:[A-Za-z0-9]+(?:=(?:[^&]*))?)?)*)?
 URIPARAM \?[A-Za-z0-9$.+!*'|(){},~@#%&/=:;_?\-\[\]]*
 URIPATHPARAM %{URIPATH}(?:%{URIPARAM})?
 URI %{URIPROTO}://(?:%{USER}(?::[^@]*)?@)?(?:%{URIHOST})?(?:%{URIPATHPARAM})?

 # Months: January, Feb, 3, 03, 12, December
 MONTH \b(?:Jan(?:uary)?|Feb(?:ruary)?|Mar(?:ch)?|Apr(?:il)?|May|Jun(?:e)?|Jul(?:y)?|
Aug(?:ust)?|Sep(?:tember)?|Oct(?:ober)?|Nov(?:ember)?|Dec(?:ember)?)\b
 MONTHNUM (?:0?[1-9]|1[0-2])
 MONTHNUM2 (?:0[1-9]|1[0-2])
 MONTHDAY (?:(?:0[1-9])|(?:[12][0-9])|(?:3[01])|[1-9])

 # Days: Monday, Tue, Thu, etc...
 DAY (?:Mon(?:day)?|Tue(?:sday)?|Wed(?:nesday)?|Thu(?:rsday)?|Fri(?:day)?|
Sat(?:urday)?|Sun(?:day)?)
```

```
 # Years?
 YEAR (?>\d\d){1,2}
 # Time: HH:MM:SS
 #TIME \d{2}:\d{2}(?::\d{2}(?:\.\d+)?)?
 # TIME %{POSINT<24}:%{POSINT<60}(?::%{POSINT<60}(?:\.%{POSINT})?)?
 HOUR (?:2[0123]|[01]?[0-9])
 MINUTE (?:[0-5][0-9])
 # '60' is a leap second in most time standards and thus is valid.
 SECOND (?:(?:[0-5]?[0-9]|60)(?:[:.,][0-9]+)?)
 TIME (?!<[0-9])%{HOUR}:%{MINUTE}(?::%{SECOND})(?![0-9])
 # datestamp is YYYY/MM/DD-HH:MM:SS.UUUU (or something like it)
 DATE_US %{MONTHNUM}[/-]%{MONTHDAY}[/-]%{YEAR}
 DATE_EU %{MONTHDAY}[./-]%{MONTHNUM}[./-]%{YEAR}
 DATESTAMP_US %{DATE_US}[- ]%{TIME}
 DATESTAMP_EU %{DATE_EU}[- ]%{TIME}
 ISO8601_TIMEZONE (?:Z|[+-]%{HOUR}(?::?%{MINUTE}))
 ISO8601_SECOND (?:%{SECOND}|60)
 TIMESTAMP_ISO8601 %{YEAR}-%{MONTHNUM}-%{MONTHDAY}[T ]%{HOUR}:?%{MINUTE}(?::?
%{SECOND})?%{ISO8601_TIMEZONE}?
 TZ (?:[PMCE][SD]T|UTC)
 DATESTAMP_RFC822 %{DAY} %{MONTH} %{MONTHDAY} %{YEAR} %{TIME} %{TZ}
 DATESTAMP_RFC2822 %{DAY}, %{MONTHDAY} %{MONTH} %{YEAR} %{TIME} %{ISO8601_TIMEZONE}
 DATESTAMP_OTHER %{DAY} %{MONTH} %{MONTHDAY} %{TIME} %{TZ} %{YEAR}
 DATESTAMP_EVENTLOG %{YEAR}%{MONTHNUM2}%{MONTHDAY}%{HOUR}%{MINUTE}%{SECOND}
 CISCOTIMESTAMP %{MONTH} %{MONTHDAY} %{TIME}

 # Syslog Dates: Month Day HH:MM:SS
 SYSLOGTIMESTAMP %{MONTH} +%{MONTHDAY} %{TIME}
 PROG (?:[\w._/%-]+)
 SYSLOGPROG %{PROG:program}(?:\[%{POSINT:pid}\])?
 SYSLOGHOST %{IPORHOST}
 SYSLOGFACILITY <%{NONNEGINT:facility}.%{NONNEGINT:priority}>
 HTTPDATE %{MONTHDAY}/%{MONTH}/%{YEAR}:%{TIME} %{INT}

 # Shortcuts
 QS %{QUOTEDSTRING:UNWANTED}

 # Log formats
 SYSLOGBASE %{SYSLOGTIMESTAMP:timestamp} (?:%{SYSLOGFACILITY} )?%{SYSLOGHOST:logsource}
 %{SYSLOGPROG}:

 MESSAGESLOG %{SYSLOGBASE} %{DATA}
```

```
 COMMONAPACHELOG %{IPORHOST:clientip} %{USER:ident} %{USER:auth}
 \[%{HTTPDATE:timestamp}\] "(?:%{WORD:verb} %{NOTSPACE:request}(?: HTTP/
%{NUMBER:httpversion})?|%{DATA:rawrequest})" %{NUMBER:response} (?:%{Bytes:bytes=
%{NUMBER}|-})
 COMBINEDAPACHELOG %{COMMONAPACHELOG} %{QS:referrer} %{QS:agent}
 COMMONAPACHELOG_DATATYPED %{IPORHOST:clientip} %{USER:ident;boolean} %{USER:auth}
 \[%{HTTPDATE:timestamp;date;dd/MMM/yyyy:HH:mm:ss Z}\] "(?:%{WORD:verb;string}
 %{NOTSPACE:request}(?: HTTP/%{NUMBER:httpversion;float})?|%{DATA:rawrequest})"
 %{NUMBER:response;int} (?:%{NUMBER:bytes;long}|-)


 # Log Levels
 LOGLEVEL ([A|a]lert|ALERT|[T|t]race|TRACE|[D|d]ebug|DEBUG|[N|n]otice|NOTICE|[I|i]nfo|
INFO|[W|w]arn?(?:ing)?|WARN?(?:ING)?|[E|e]rr?(?:or)?|ERR?(?:OR)?|[C|c]rit?(?:ical)?|
CRIT?(?:ICAL)?|[F|f]atal|FATAL|[S|s]evere|SEVERE|EMERG(?:ENCY)?|[Ee]merg(?:ency)?)
```

Escritura de clasificadores personalizados XML

XML define la estructura de un documento utilizando etiquetas en el archivo. Con un clasificador personalizado XML, puede especificar el nombre de etiqueta usado para definir una fila.

Valores personalizados del clasificador en AWS Glue

Al definir un clasificador XML, proporciona los siguientes valores a AWS Glue para crear el clasificador. El campo de clasificación de este clasificador se establece en `xml`.

Nombre

Nombre del clasificador.

Etiqueta de fila

El nombre de etiqueta XML que define una fila de tabla en el documento XML, sin paréntesis < >. El nombre debe cumplir las reglas de XML para una etiqueta.

> **Note**
>
> El elemento que contiene los datos de fila no puede ser un elemento vacío de autocierre. Por ejemplo, noAWS Glue analiza este elemento vacío:
>
> ```
> <row att1=”xx” att2=”yy” />
> ```

Los elementos vacíos se pueden escribir de la siguiente manera:

```
<row att1=”xx” att2=”yy”> </row>
```

AWS Glue realiza un seguimiento de la hora de creación, la hora de la última actualización y la versión de su clasificador.

Por ejemplo, suponga que tiene el archivo XML siguiente. Para crear una tabla de AWS Glue que solo contiene columnas para autor y título, cree un clasificador en la consola de AWS Glue con Row tag (Etiqueta de fila) como `AnyCompany`. A continuación, añada y ejecute un rastreador que use este clasificador personalizado.

```
<?xml version="1.0"?>
<catalog>
   <book id="bk101">
     <AnyCompany>
       <author>Rivera, Martha</author>
       <title>AnyCompany Developer Guide</title>
     </AnyCompany>
   </book>
   <book id="bk102">
     <AnyCompany>
       <author>Stiles, John</author>
       <title>Style Guide for AnyCompany</title>
     </AnyCompany>
   </book>
</catalog>
```

Escritura de clasificadores personalizados JSON

JSON es un formato de intercambio de datos. Define estructuras de datos con pares nombre-valor o una lista ordenada de valores. Con un clasificador personalizado JSON, puede especificar la ruta de JSON a una estructura de datos que se usa para definir el esquema para su tabla.

Valores personalizados del clasificador en AWS Glue

Al definir un clasificador JSON, proporciona los siguientes valores a AWS Glue para crear el clasificador. El campo de clasificación de este clasificador se establece en `json`.

Nombre

Nombre del clasificador.

JSON path (Ruta JSON)

Ruta de JSON que apunta a un objeto que se usa para definir un esquema de tabla. La ruta de JSON se puede escribir en la notación de puntos o la notación de corchete. Se admiten los siguientes operadores:

Descripción
Elemento raíz de un objeto JSON. Esto inicia todas las expresiones de ruta
Carácter comodín. Disponible allí donde se requiera un nombre o un valor numérico en la ruta de JSON.
Elemento secundario con notación de puntos. Especifica un campo secundario en un objeto JSON.
Elemento secundario con notación de corchete. Especifica el campo secundario en un objeto JSON. Solo se puede especificar un único campo secundario.
Índice de matriz. Especifica el valor de una matriz por índice.

AWS Glue realiza un seguimiento de la hora de creación, la hora de la última actualización y la versión de su clasificador.

Example Uso de un clasificador JSON para extraer registros de una matriz

Supongamos que sus datos JSON son una matriz de registros. Por ejemplo, las primeras líneas de su archivo podrían tener el siguiente aspecto:

```
[
```

```
  {
    "type": "constituency",
    "id": "ocd-division\/country:us\/state:ak",
    "name": "Alaska"
  },
  {
    "type": "constituency",
    "id": "ocd-division\/country:us\/state:al\/cd:1",
    "name": "Alabama's 1st congressional district"
  },
  {
    "type": "constituency",
    "id": "ocd-division\/country:us\/state:al\/cd:2",
    "name": "Alabama's 2nd congressional district"
  },
  {
    "type": "constituency",
    "id": "ocd-division\/country:us\/state:al\/cd:3",
    "name": "Alabama's 3rd congressional district"
  },
  {
    "type": "constituency",
    "id": "ocd-division\/country:us\/state:al\/cd:4",
    "name": "Alabama's 4th congressional district"
  },
  {
    "type": "constituency",
    "id": "ocd-division\/country:us\/state:al\/cd:5",
    "name": "Alabama's 5th congressional district"
  },
  {
    "type": "constituency",
    "id": "ocd-division\/country:us\/state:al\/cd:6",
    "name": "Alabama's 6th congressional district"
  },
  {
    "type": "constituency",
    "id": "ocd-division\/country:us\/state:al\/cd:7",
    "name": "Alabama's 7th congressional district"
  },
  {
    "type": "constituency",
    "id": "ocd-division\/country:us\/state:ar\/cd:1",
    "name": "Arkansas's 1st congressional district"
```

```
  },
  {
    "type": "constituency",
    "id": "ocd-division\/country:us\/state:ar\/cd:2",
    "name": "Arkansas's 2nd congressional district"
  },
  {
    "type": "constituency",
    "id": "ocd-division\/country:us\/state:ar\/cd:3",
    "name": "Arkansas's 3rd congressional district"
  },
  {
    "type": "constituency",
    "id": "ocd-division\/country:us\/state:ar\/cd:4",
    "name": "Arkansas's 4th congressional district"
  }
]
```

Al ejecutar un rastreador mediante el clasificador JSON integrado, todo el archivo se usará para definir el esquema. Como no especifica una ruta de JSON, el rastreador tratará los datos como un objeto, es decir, solo una matriz. Por ejemplo, el esquema podría tener el siguiente aspecto:

```
root
|-- record: array
```

Sin embargo, para crear un esquema que se basa en cada registro de la matriz JSON, cree un clasificador JSON personalizado y especifique la ruta de JSON como `$[*]`. Al especificar esta ruta de JSON, el clasificador interroga a los 12 registros de la matriz para determinar el esquema. El esquema resultante contiene campos independientes para cada objeto, similares al siguiente ejemplo:

```
root
|-- type: string
|-- id: string
|-- name: string
```

Example Uso de un clasificador JSON para examinar solo las partes de un archivo

Supongamos que sus datos JSON siguen el patrón del archivo JSON de ejemplo `s3://awsglue-datasets/examples/us-legislators/all/areas.json` procedente de http://everypolitician.org/. Los objetos de ejemplo del archivo JSON tienen un aspecto similar al siguiente:

```
{
  "type": "constituency",
  "id": "ocd-division\/country:us\/state:ak",
  "name": "Alaska"
}
{
  "type": "constituency",
  "identifiers": [
    {
      "scheme": "dmoz",
      "identifier": "Regional\/North_America\/United_States\/Alaska\/"
    },
    {
      "scheme": "freebase",
      "identifier": "\/m\/0hjy"
    },
    {
      "scheme": "fips",
      "identifier": "US02"
    },
    {
      "scheme": "quora",
      "identifier": "Alaska-state"
    },
    {
      "scheme": "britannica",
      "identifier": "place\/Alaska"
    },
    {
      "scheme": "wikidata",
      "identifier": "Q797"
    }
  ],
  "other_names": [
    {
      "lang": "en",
```

```
      "note": "multilingual",
      "name": "Alaska"
    },
    {
      "lang": "fr",
      "note": "multilingual",
      "name": "Alaska"
    },
    {
      "lang": "nov",
      "note": "multilingual",
      "name": "Alaska"
    }
  ],
  "id": "ocd-division\/country:us\/state:ak",
  "name": "Alaska"
}
```

Al ejecutar un rastreador mediante el clasificador JSON integrado, todo el archivo se usará para crear el esquema. Es posible que termine contando con un esquema similar a este:

```
root
|-- type: string
|-- id: string
|-- name: string
|-- identifiers: array
|    |-- element: struct
|    |    |-- scheme: string
|    |    |-- identifier: string
|-- other_names: array
|    |-- element: struct
|    |    |-- lang: string
|    |    |-- note: string
|    |    |-- name: string
```

Sin embargo, para crear un esquema con solo el objeto "`id`", cree un clasificador JSON personalizado y especifique la ruta de JSON como `$.id`. A continuación, el esquema se basa solo en el campo "`id`":

```
root
|-- record: string
```

Las primeras líneas de datos extraídas con este esquema tienen el siguiente aspecto:

```
{"record": "ocd-division/country:us/state:ak"}
{"record": "ocd-division/country:us/state:al/cd:1"}
{"record": "ocd-division/country:us/state:al/cd:2"}
{"record": "ocd-division/country:us/state:al/cd:3"}
{"record": "ocd-division/country:us/state:al/cd:4"}
{"record": "ocd-division/country:us/state:al/cd:5"}
{"record": "ocd-division/country:us/state:al/cd:6"}
{"record": "ocd-division/country:us/state:al/cd:7"}
{"record": "ocd-division/country:us/state:ar/cd:1"}
{"record": "ocd-division/country:us/state:ar/cd:2"}
{"record": "ocd-division/country:us/state:ar/cd:3"}
{"record": "ocd-division/country:us/state:ar/cd:4"}
{"record": "ocd-division/country:us/state:as"}
{"record": "ocd-division/country:us/state:az/cd:1"}
{"record": "ocd-division/country:us/state:az/cd:2"}
{"record": "ocd-division/country:us/state:az/cd:3"}
{"record": "ocd-division/country:us/state:az/cd:4"}
{"record": "ocd-division/country:us/state:az/cd:5"}
{"record": "ocd-division/country:us/state:az/cd:6"}
{"record": "ocd-division/country:us/state:az/cd:7"}
```

Para crear un esquema basado en un objeto profundamente anidado, como "`identifier`", en el archivo JSON, puede crear un clasificador JSON personalizado y especificar la ruta de JSON como `$.identifiers[*].identifier`. Aunque el esquema es similar al ejemplo anterior, se basa en un objeto diferente del archivo JSON.

El esquema sería similar al siguiente:

```
root
|-- record: string
```

Al mostrar las primeras líneas de datos de la tabla, se muestra a su vez que el esquema se basa en los datos del objeto "`identifier`":

```
{"record": "Regional/North_America/United_States/Alaska/"}
{"record": "/m/0hjy"}
{"record": "US02"}
{"record": "5879092"}
{"record": "4001016-8"}
{"record": "destination/alaska"}
{"record": "1116270"}
{"record": "139487266"}
{"record": "n79018447"}
{"record": "01490999-8dec-4129-8254-eef6e80fadc3"}
{"record": "Alaska-state"}
{"record": "place/Alaska"}
{"record": "Q797"}
{"record": "Regional/North_America/United_States/Alabama/"}
{"record": "/m/0gyh"}
{"record": "US01"}
{"record": "4829764"}
{"record": "4084839-5"}
{"record": "161950"}
{"record": "131885589"}
```

Para crear una tabla basada en otro objeto profundamente anidado, como el campo "`name`" de la matriz "`other_names`" del archivo JSON, puede crear un clasificador JSON personalizado y especificar la ruta de JSON como `$.other_names[*].name`. Aunque el esquema es similar al ejemplo anterior, se basa en un objeto diferente del archivo JSON. El esquema sería similar al siguiente:

```
root
|-- record: string
```

Al mostrar las primeras líneas de datos de la tabla, se muestra a su vez que se basa en los datos del objeto "`name`" de la matriz "`other_names`":

```
{"record": "Alaska"}
{"record": "Alaska"}
{"record": "Аляска"}
{"record": "Alaska"}
{"record": "Alaska"}
{"record": "Alaska"}
{"record": "Alaska"}
```

```
{"record": "Alaska"}
{"record": "Alaska"}
{"record": "######"}
{"record": "######"}
{"record": "######"}
{"record": "Alaska"}
{"record": "Alyaska"}
{"record": "Alaska"}
{"record": "Alaska"}
{"record": "Штат Аляска"}
{"record": "Аляска"}
{"record": "Alaska"}
{"record": "#######"}
```

Escritura de clasificadores personalizados CSV

Los clasificadores CSV personalizados permiten especificar los tipos de datos para cada columna en el campo del clasificador CSV personalizado. Se puede especificar el tipo de datos separados por una coma. Al especificar los tipos de datos, se pueden anular los tipos de datos inferidos por los rastreadores y garantizar que los datos se clasifiquen correctamente.

Puede configurar el SerDe para procesar CSV en el clasificador, lo que se aplicará en el Data Catalog.

Al crear un clasificador personalizado, también se puede volver a utilizarlo para diferentes rastreadores.

- Para los archivos CSV con solo encabezados (sin datos), estos archivos se clasificarán como DESCONOCIDOS ya que no se proporciona suficiente información. Si especifica que el CSV "tiene encabezados" en la opción Encabezados de columna y proporciona los tipos de datos, se pueden clasificar estos archivos correctamente.

Puede utilizar un clasificador CSV personalizado para deducir el esquema de distintos tipos de datos CSV. Entre los atributos personalizados que puede proporcionar para el clasificador, se incluyen los delimitadores, una opción de SerDe de CSV, las opciones sobre el encabezado y si se van a realizar o no ciertas validaciones de los datos.

Valores personalizados del clasificador en AWS Glue

Cuando defina un clasificador CSV, debe proporcionar los siguientes valores a AWS Glue para que pueda crear el clasificador. El campo de clasificación de este clasificador se establece en `csv`.

Classifier name (Nombre de clasificador)

Nombre del clasificador.

Serde de CSV

Establece el SerDe para procesar CSV en el clasificador, que se aplicará en el catálogo de datos. Las opciones son Open CSV SerDe, Lazy Simple SerDe y None. Puede especificar el valor Ninguno cuando desee que el rastreador realice la detección.

Delimitador de columnas

Símbolo personalizado que indica qué elemento va a separar cada entrada de columna en la fila. Proporcione un carácter Unicode. Si no puede escribir el delimitador, puede copiarlo y pegarlo. Esto funciona con los caracteres imprimibles, incluidos con los que el sistema no es compatible (normalmente se muestran como □).

Símbolo de comillas

Símbolo personalizado que indica qué elemento va a combinar contenido en un valor de columna único. Debe ser diferente al delimitador de columnas. Proporcione un carácter Unicode. Si no puede escribir el delimitador, puede copiarlo y pegarlo. Esto funciona con los caracteres imprimibles, incluidos con los que el sistema no es compatible (normalmente se muestran como □).

Encabezados de columna

Indica cómo deben detectarse los encabezados de columna en el archivo CSV. Si el archivo CSV personalizado tiene encabezados de columna, escriba una lista delimitada por comas con los encabezados de columna.

Opciones de procesamiento: permitir archivos con una sola columna

Permite procesar los archivos que contienen una sola columna.

Opciones de procesamiento: quitar un espacio en blanco antes de identificar los valores de columna

Indica si los valores se van a recortar antes de identificar el tipo de valores de columna.

Tipos de datos personalizados: opcional

Ingrese el tipo de datos personalizado separados por una coma. Especifica los tipos de datos personalizados del archivo CSV. El tipo de datos personalizado debe ser un tipo de datos compatible. Los tipos de datos admitidos son: "BINARIO", "BOOLEANO", "FECHA", "DECIMAL", "DOUBLE", "FLOAT", "INT", "LONG", "SHORT", "STRING", "TIMESTAMP". Los tipos de datos no compatibles mostrarán un error.

Trabajo con clasificadores en la consola de AWS Glue

Un clasificador determina el esquema de sus datos. Puede escribir un clasificador personalizado y apuntar a él desde AWS Glue.

Visualización de clasificadores

Para ver una lista de todos los clasificadores que ha creado, abra la consola de AWS Glue en https://console.aws.amazon.com/glue/ y elija la pestaña Classifiers (Clasificadores).

La lista muestra las siguientes propiedades sobre cada clasificador:

- Clasificadores: nombre del clasificador. Cuando crea un clasificador, debe proporcionarle un nombre.
- Clasificación: tipo de clasificación de las tablas que este clasificador deduce.
- Última actualización: última actualización de este clasificador.

Administración de clasificadores

En la lista Classifiers (Clasificadores) de la consola de AWS Glue, puede añadir, editar y eliminar clasificadores. Para consultar más detalles sobre un clasificador, seleccione el nombre del clasificador en la lista. Los detalles incluirán la información que definió al crear el clasificador.

Creación de clasificadores

Para añadir un clasificador en la consola de AWS Glue, seleccione Add classifier (Añadir clasificador). Cuando define un clasificador, suministra valores para los elementos siguientes:

- Nombre del clasificador: facilita un nombre único para el clasificador.
- Tipo de clasificador: tipo de clasificación de las tablas que este clasificador deduce.
- Última actualización: última actualización de este clasificador.

Classifier name (Nombre de clasificador)

Facilite un nombre único para el clasificador.

Tipo de clasificador

Elija el tipo de clasificador que debe crearse.

Dependiendo del tipo de clasificador que elija, configure las siguientes propiedades para el clasificador:

Grok

- Clasificación

  Describa el formato o el tipo de datos que se clasifica o proporcione una etiqueta personalizada.

- Patrón de Grok

  Este valor se utiliza para analizar los datos en un esquema estructurado. El patrón de grok se compone de patrones con nombre que describen el formato de su almacén de datos. Debe escribir este patrón de grok con los patrones integrados con nombre que AWS Glue proporciona y los patrones personalizados que escribe e incluye en el campo Custom patterns (Patrones personalizados). Aunque puede que los resultados del depurador de grok no coincidan exactamente con los resultados de AWS Glue, le sugerimos que pruebe su patrón utilizando algunos datos de muestra con un depurador de grok. Puede encontrar depuradores de grok en la web. Los patrones integrados con nombre que AWS Glue proporciona suelen ser compatibles con los patrones de grok que están disponibles en la web.

  Cree su patrón de grok añadiendo iterativamente patrones con nombre y compruebe sus resultados en un depurador. Esta actividad le ofrece la confianza de que cuando el rastreador de AWS Glue ejecute el patrón de grok, sus datos se podrán analizar.

- Patrones personalizados

  Para los clasificadores de grok, son componentes básicos opcionales del Grok pattern (Patrón de grok) que escribe. Cuando los patrones integrados no pueden analizar sus datos, es posible que tenga que escribir un patrón personalizado. Estos patrones personalizados se definen en este campo y se hace referencia a ellos en el campo Grok pattern (Patrón de Grok). Cada patrón personalizado se define en una línea independiente. Al igual que los patrones integrados, se compone de una definición de patrón con nombre que utiliza la sintaxis expresión regular (regex).

  En el ejemplo siguiente se ve el nombre `MESSAGEPREFIX` seguido de una definición de expresión regular para aplicarla a sus datos a fin de establecer si siguen el patrón o no.

```
MESSAGEPREFIX .*-.*-.*-.*-.*
```

XML

- Etiqueta de fila

  Para los clasificadores de XML, nombre de la etiqueta XML que define una fila de una tabla en el documento XML. Escriba el nombre sin corchetes angulares < >. El nombre debe cumplir las reglas de XML para una etiqueta.

  Para obtener más información, consulte Escritura de clasificadores personalizados XML.

JSON

- JSON path (Ruta JSON)

  Para los clasificadores de JSON, ruta de JSON al objeto, la matriz o el valor que define una fila de la tabla que se está creando. Escriba el nombre con sintaxis JSON de punto o corchete y utilizando los operadores compatibles de AWS Glue.

  Para obtener más información, consulte la lista de operadores en Escritura de clasificadores personalizados JSON.

CSV

- Delimitador de columnas

  Único caracter o símbolo personalizado que indica qué elemento va a separar cada entrada de columna en la fila. Elija el delimitador en la lista o elija `Other` para introducir un delimitador personalizado.

- Símbolo de comillas

  Único caracter o símbolo personalizado que indica qué elemento va a combinar contenido en un valor de columna único. Debe ser diferente al delimitador de columnas. Elija el símbolo de comillas de la lista o elija `Other` para introducir un carácter de comilla personalizado.

- Encabezados de columna

Indica cómo deben detectarse los encabezados de columna en el archivo CSV. Puede elegir `Has headings`, `No headings`, o `Detect headings`. Si el archivo CSV personalizado tiene encabezados de columna, escriba una lista delimitada por comas con los encabezados de columna.

- Permita archivos con una sola columna

  Para clasificarse como CSV, los datos deben tener al menos dos columnas y dos filas de datos. Utilice esta opción para permitir procesar los archivos que contienen una sola columna.

- Quite los espacios en blanco antes de identificar los valores de columna

  Esta opción indica si los valores se van a recortar antes de identificar el tipo de valores de columna.

- Tipo de datos personalizado

  (Opcional): ingrese los tipos de datos personalizados en una lista delimitada por comas. Los tipos de datos admitidos son: "BINARIO", "BOOLEANO", "FECHA", "DECIMAL", "DOUBLE", "FLOAT", "INT", "LONG", "SHORT", "STRING", "TIMESTAMP".

- Serde de CSV

  (Opcional): un SerDe para procesar CSV en el clasificador, lo que se aplicará en el catálogo de datos. Elija entre `Open CSV SerDe`, `Lazy Simple SerDe` o `None`. Puede especificar el valor `None` cuando desee que el rastreador realice la detección.

Para obtener más información, consulte Escritura de clasificadores personalizados.

Programación de un rastreador de AWS Glue

Puede ejecutar un rastreador de AWS Glue bajo demanda o según una programación periódica. Las programaciones del rastreador se pueden expresar en formato Cron. Para obtener más información, consulte Cron en Wikipedia.

Al crear un rastreador según una programación, puede especificar determinadas restricciones, como la frecuencia, los días de la semana y la hora en que se ejecuta el rastreador. Estas restricciones se basan en Cron. Al configurar una programación de rastreador, debe tener en cuenta las características y limitaciones de Cron. Por ejemplo, si decide ejecutar su rastreador el día 31 de cada mes, tenga en cuenta que algunos meses no tienen 31 días.

Los rastreos de cada rastreador solo son válidos durante un máximo de 12 meses

Para obtener más información acerca cómo utilizar Cron para programar trabajos y rastreadores, consulte Programaciones basadas en tiempo para trabajos y rastreadores.

Ver los resultados y detalles del rastreador

Una vez que el rastreador se ejecuta correctamente, crea definiciones de tabla en el Catálogo de datos. Elija Tables (Tablas) en el panel de navegación para ver las tablas que creó su rastreador en la base de datos especificada.

Puede ver la información relacionada con el rastreador en sí de la siguiente manera:

- La página Crawlers (Rastreadores) en la consola de AWS Glue muestra las siguientes propiedades de un rastreador:

Propiedad	Descripción
Nombre	Cuando se crea un rastreador, se le debe asignar un nombre único.
Status	Un rastreador puede tener los siguientes estados: ready, starting, stopping, scheduled o schedule paused. Un rastreador en ejecución progresa de starting a stopping. Puede reanudar o pausar una programación asociada a un rastreador.
Programación	Puede elegir ejecutar su rastreador bajo demanda o elegir una frecuencia con una programación. Para obtener más información acerca de la programación de un rastreador, consulte Programación de un rastreador.
Last run (Última ejecución)	Fecha y hora en que se ejecutó el rastreador por última vez.
Log (Registro)	Enlaces a cualquier registro disponible de la última ejecución del rastreador.

Propiedad	Descripción
Tables changes from last run (Cambios en las tablas desde la última ejecución)	El número de tablas en el AWS Glue Data Catalog que actualizó la última ejecución del rastreador.

- Para el historial de un rastreador, elija Crawlers (Rastreadores) en el panel de navegación con el fin de ver los rastreadores que haya creado. Elija un rastreador de la lista de rastreadores disponibles. Puede ver las propiedades y el historial del rastreador en la pestaña Crawler runs (Ejecuciones del rastreador).

  La pestaña Crawler runs (Ejecuciones del rastreador) muestra información sobre cada una de las veces que se ha ejecutado el rastreador, incluyendo Start time (UTC) (Hora de inicio [UTC]), End time (UTC) (Hora de finalización [UTC]), Duration (Duración), Status (Estado), DPU hours (Horas de DPU) y Table changes (Cambios en las tablas).

  La pestaña del ejecutor del rastreador muestra solo los rastreos que se han producido desde la fecha de lanzamiento de la función de historial del rastreador y solo conserva hasta 12 meses de rastreo. No se devolverán los rastreos más antiguos.
- Para ver información adicional, elija una pestaña en la página de detalles del rastreador. En cada pestaña se mostrará información relacionada con el rastreador.
  - Schedule (Programación): aquí aparecerán los programas creados para el rastreador.
  - Data sources (Orígenes de datos): aquí aparecerán todos los orígenes de datos analizados por el rastreador.
  - Classifiers (Clasificadores): aquí aparecerán todos los clasificadores asignados al rastreador.
  - Tags (Etiquetas): aquí aparecerán las etiquetas creadas y asignadas a un recurso de AWS.

Parámetros establecidos en las tablas del catálogo de datos por el rastreador

Los rastreadores de AWS Glue establecen estas propiedades de la tabla. Esperamos que los usuarios consuman las propiedades `classification` y `compressionType`. Para los cálculos internos se utilizan otras propiedades, incluidas las estimaciones del tamaño de las tablas, y no se garantizan su precisión ni aplicabilidad a los casos de uso de los clientes. Cambiar estos parámetros puede alterar el comportamiento del rastreador y no admitimos este flujo de trabajo.

Clave de la propiedad	Valor de la propiedad
`UPDATED_BY_CRAWLER`	Nombre del rastreador que realiza la actualización.
`connectionName`	El nombre de la conexión del Catálogo de datos para el rastreador que se utiliza para la conexión al almacén de datos.
`recordCount`	Calcule el recuento de registros de la tabla, según el tamaño de los archivos y los encabezados.
`skip.header.line.count`	Filas omitidas para omitir el encabezado. Se establece en tablas clasificadas como CSV.
`CrawlerSchemaSerializerVersion`	Para uso interno
`classification`	Formato de los datos, deducido por el rastreador. Para obtener más información sobre los formatos de datos que admiten los rastreadores de AWS Glue, consulte the section called “Clasificadores integrados en AWS Glue”.
`CrawlerSchemaDeserializerVersion`	Para uso interno
`sizeKey`	Tamaño combinado de los archivos de la tabla rastreada.
`averageRecordSize`	Tamaño promedio de una fila en la tabla, en bytes.
`compressionType`	Tipo de compresión utilizado en los datos de la tabla. Para obtener más información sobre los tipos de compresión que admiten los rastreadores de AWS Glue, consulte the section called “Clasificadores integrados en AWS Glue”.
`typeOfData`	`file`, `table` o bien `view`.

Clave de la propiedad	Valor de la propiedad
`objectCount`	Número de objetos en la ruta de Amazon S3 para la tabla.

Los rastreadores de AWS Glue establecen estas propiedades de tabla adicionales para almacenes de datos de Snowflake.

Clave de la propiedad	Valor de la propiedad
`aws:RawTableLastAltered`	Registra la última marca temporal modificada de la tabla de Snowflake.
`ViewOriginalText`	Vea la instrucción SQL.
`ViewExpandedText`	Vea la instrucción SQL codificada en formato Base64.
`ExternalTable:S3Location`	Ubicación de Amazon S3 de la tabla externa de Snowflake.
`ExternalTable:FileFormat`	Formato de archivo de Amazon S3 de la tabla externa de Snowflake.

Los rastreadores de AWS Glue establecen estas propiedades de tabla adicionales para almacenes de datos de tipo JDBC, como Amazon Redshift, Microsoft SQL Server, MySQL, PostgreSQL y Oracle.

Clave de la propiedad	Valor de la propiedad
`aws:RawType`	Cuando un rastreador almacena los datos en el Catálogo de datos, traduce los tipos de datos a tipos compatibles con Hive, lo que muchas veces hace que se pierda la información del tipo de datos nativo. El

Clave de la propiedad	Valor de la propiedad
	rastreador genera el parámetro `aws:RawType` para proporcionar el tipo de datos de nivel nativo.
`aws:RawColumnComment`	Si un comentario está asociado a una columna de la base de datos, el rastreador genera el comentario correspondiente en la tabla del catálogo. La cadena de comentario se trunca en 255 bytes. Microsoft SQL Server no admite los comentarios.
`aws:RawTableComment`	Si un comentario está asociado a una tabla de la base de datos, el rastreador genera el comentario correspondiente en la tabla del catálogo. La cadena de comentario se trunca en 255 bytes. Microsoft SQL Server no admite los comentarios.

Personalización del comportamiento del rastreador

Cuando un rastreador se ejecuta, puede encontrar cambios en su almacén de datos que dan lugar a un esquema o una partición que es diferente de un rastreo anterior. Puede utilizar la AWS Management Console o la API de AWS Glue para configurar la manera en que su rastreador procesa determinados tipos de cambios.

Temas

- Rastreos progresivos para agregar nuevas particiones
- Establecer la opción de configuración del rastreador de índices de particiones
- Aceleración de los rastreadores mediante las notificaciones de eventos de Amazon S3
- Cómo evitar que el rastreador cambie un esquema existente
- Cómo crear un único esquema para cada ruta de inclusión de Amazon S3
- Cómo especificar la ubicación de la tabla y el nivel de partición
- Cómo especificar el número máximo de tablas que el rastreador tiene permitido crear
- Cómo especificar opciones de configuración para un almacén de datos de Delta Lake
- Cómo configurar un rastreador para que utilice credenciales de Lake Formation

Console

Al definir un rastreador mediante la consola de AWS Glue, dispone de varias opciones para configurar el comportamiento de su rastreador. Para obtener más información acerca de cómo usar la consola de AWS Glue para añadir un rastreador, consulte Configuración de rastreadores.

Cuando un rastreador se ejecuta en un almacén de datos rastreado anteriormente, puede resultar que un esquema haya cambiado o que se hayan eliminado objetos del almacén de datos. El rastreador registra los cambios de un esquema. En función del tipo de fuente del rastreador, las particiones y las tablas nuevas se podrían crear independientemente de la política de cambio de esquema.

Para especificar lo que hace el rastreador cuando detecta modificaciones en el esquema, puede elegir una de las siguientes acciones en la consola:

- Actualizar la definición de la tabla en el Catálogo de datos: agregar columnas nuevas, eliminar las que faltan y modificar las definiciones de las columnas existentes en el AWS Glue Data Catalog. Eliminar los metadatos que no haya configurado el rastreador. Este es el valor predeterminado.
- Agregar solo columnas nuevas: en el caso de las tablas que se mapean a un almacén de datos de Amazon S3, agrega columnas nuevas a medida que se detectan, pero no se quita ni cambia el tipo de las columnas existentes en el Catálogo de datos. Elija esta opción cuando las columnas actuales del Catálogo de datos sean correctas y no desea que el rastreador quite ni cambie el tipo de las columnas existentes. Si cambia un atributo de tabla de Amazon S3 fundamental, como clasificación, tipo de compresión o delimitador CSV, marque la tabla como obsoleta. Mantenga el formato de entrada y de salida, tal como se encuentra en el Catálogo de datos. Actualice los parámetros de SerDe únicamente si el parámetro es uno de los definidos por el rastreador. En los demás almacenes de datos, modifique las definiciones de columna existentes.
- Ignorar el cambio y no actualizar la tabla en el Catálogo de datos: solo se crean tablas y particiones nuevas.

  Esta es la configuración predeterminada para rastreos progresivos.

Un rastreador también podría descubrir particiones nuevas o modificadas. De forma predeterminada, se añaden nuevas particiones y se actualizan particiones existentes si han cambiado. Además, puede establecer una opción de configuración del rastreador en Actualizar todas las particiones nuevas y existentes con metadatos de la tabla en la consola de AWS

Glue. Al establecerse esta opción, las particiones heredan propiedades de metadatos, como su clasificación, formato de entrada, formato de salida, información de SerDe y esquema, de su tabla principal. Cualquier cambio en esas propiedades de una tabla se propaga por sus particiones. Cuando esta opción de configuración se establece en un rastreador existente, las particiones existentes se actualizan para que coincidan con las propiedades de su tabla principal la próxima vez que se ejecuta el rastreador.

Para especificar lo que hace el rastreador cuando encuentra un objeto eliminado en el almacén de datos, elija una de las siguientes acciones:

- Eliminar tablas y particiones del Catálogo de datos
- Ignorar el cambio y no actualizar la tabla en el Catálogo de datos

  Esta es la configuración predeterminada para rastreos progresivos.
- Marcar la tabla como obsoleta en el Catálogo de datos: esta es la configuración predeterminada.

AWS CLI

```
aws glue create-crawler \
--name "your-crawler-name" \
--role "your-iam-role-arn" \
--database-name "your-database-name" \
--targets 'S3Targets=[{Path="s3://your-bucket-name/path-to-data"}]' \
--configuration '{"Version": 1.0, "CrawlerOutput": {"Partitions":
 {"AddOrUpdateBehavior": "InheritFromTable"}, "Tables": {"AddOrUpdateBehavior":
 "MergeNewColumns"}}}'
```

API

Al definir un rastreador mediante la API de AWS Glue, puede elegir entre varios campos para configurar el rastreador. `SchemaChangePolicy` en la API del rastreador determina qué hace este cuando descubre un esquema cambiado o un objeto eliminado. El rastreador registra cambios de esquema a medida que se ejecuta.

Ejemplo de un código de Python con las opciones de configuración de rastreador

```
import boto3
import json
```

```
# Initialize a boto3 client for AWS Glue
glue_client = boto3.client('glue', region_name='us-east-1')  # Replace 'us-east-1'
 with your desired AWS region

# Define the crawler configuration
crawler_configuration = {
    "Version": 1.0,
    "CrawlerOutput": {
        "Partitions": {
            "AddOrUpdateBehavior": "InheritFromTable"
        },
        "Tables": {
            "AddOrUpdateBehavior": "MergeNewColumns"
        }
    }
}

configuration_json = json.dumps(crawler_configuration)
# Create the crawler with the specified configuration
response = glue_client.create_crawler(
    Name='your-crawler-name',  # Replace with your desired crawler name
    Role='crawler-test-role',  # Replace with the ARN of your IAM role for Glue
    DatabaseName='default',  # Replace with your target Glue database name
    Targets={
        'S3Targets': [
            {
                'Path': "s3://your-bucket-name/path/",  # Replace with your S3 path
 to the data
            },
        ],
        # Include other target types like 'JdbcTargets' if needed
    },
    Configuration=configuration_json,
    # Include other parameters like Schedule, Classifiers, TablePrefix,
 SchemaChangePolicy, etc., as needed
)

print(response)a
```

Al ejecutarse un rastreador, siempre se crean nuevas tablas y particiones independientemente de la política de cambio de esquema. Puede elegir una de las acciones siguientes en el campo `UpdateBehavior` de la estructura `SchemaChangePolicy` para determinar lo que hace el rastreador cuando encuentra un esquema de tabla cambiado:

- `UPDATE_IN_DATABASE`: actualizar la tabla en AWS Glue Data Catalog. Agregar columnas nuevas, eliminar las que faltan y modificar las definiciones de las columnas existentes. Eliminar los metadatos que no haya configurado el rastreador.
- `LOG`: ignorar el cambio y no actualizar la tabla en el Catálogo de datos.

  Esta es la configuración predeterminada para rastreos progresivos.

También puede sustituir la estructura `SchemaChangePolicy` mediante un objeto JSON proporcionado en el campo `Configuration` de la API del rastreador. Este objeto JSON puede contener un par clave-valor para definir la política de modo que no actualice las columnas existentes y solo agregue las nuevas. Por ejemplo, proporcione el siguiente objeto JSON como una cadena:

```
{
   "Version": 1.0,
   "CrawlerOutput": {
      "Tables": { "AddOrUpdateBehavior": "MergeNewColumns" }
   }
}
```

Esta opción corresponde a la opción Agregar solo columnas nuevas en la consola de AWS Glue. Sustituye la estructura `SchemaChangePolicy` de las tablas que se derivan solo del rastreo de los almacenes de datos de Amazon S3. Seleccione esta opción si desea mantener los metadatos tal como están en el Catálogo de datos (el origen de confianza). Las columnas nuevas se agregan a medida que se encuentran, incluidos los tipos de datos anidados. Sin embargo, las columnas existentes no se eliminan y su tipo no cambia. Si un atributo de tabla de Amazon S3 cambia de forma significativa, la tabla se marca como obsoleta y se registra una advertencia de que un atributo incompatible se debe resolver. Esta opción no se aplica a los rastreadores incrementales.

Cuando un rastreador se ejecuta en un almacén de datos rastreado anteriormente, puede resultar que se detecten particiones nuevas o cambiadas. De forma predeterminada, se añaden nuevas particiones y se actualizan particiones existentes si han cambiado. Además, puede establecer

una opción de configuración del rastreador en `InheritFromTable` (correspondiente a la opción Actualizar todas las particiones nuevas y existentes con metadatos de la tabla en la consola de AWS Glue). Al establecerse esta opción, las particiones heredan propiedades de metadatos de su tabla principal, como su clasificación, formato de entrada, formato de salida, información del SerDe y esquema. Cualquier cambio en las propiedades a la tabla principal se propaga por sus particiones.

Cuando esta opción de configuración se establece en un rastreador existente, las particiones existentes se actualizan para que coincidan con las propiedades de su tabla principal la próxima vez que se ejecuta el rastreador. Este comportamiento se establece en el campo `Configuration` de la API del rastreador. Por ejemplo, proporcione el siguiente objeto JSON como una cadena:

```
{
   "Version": 1.0,
   "CrawlerOutput": {
      "Partitions": { "AddOrUpdateBehavior": "InheritFromTable" }
   }
}
```

El campo `Configuration` de la API del rastreador puede establecer varias opciones de configuración. Por ejemplo, para configurar la salida del rastreador tanto para particiones como para tablas, puede proporcionar una representación de cadena del siguiente objeto JSON:

```
{
    "Version": 1.0,
    "CrawlerOutput": {
       "Partitions": { "AddOrUpdateBehavior": "InheritFromTable" },
       "Tables": {"AddOrUpdateBehavior": "MergeNewColumns" }
    }
}
```

Puede elegir una de las acciones siguientes para determinar lo que hace el rastreador cuando encuentra un objeto eliminado en el almacén de datos. El campo `DeleteBehavior` de la estructura `SchemaChangePolicy` en la API del rastreador establece el comportamiento del rastreador cuando descubre un objeto eliminado.

- `DELETE_FROM_DATABASE`: eliminar tablas y particiones del Catálogo de datos.

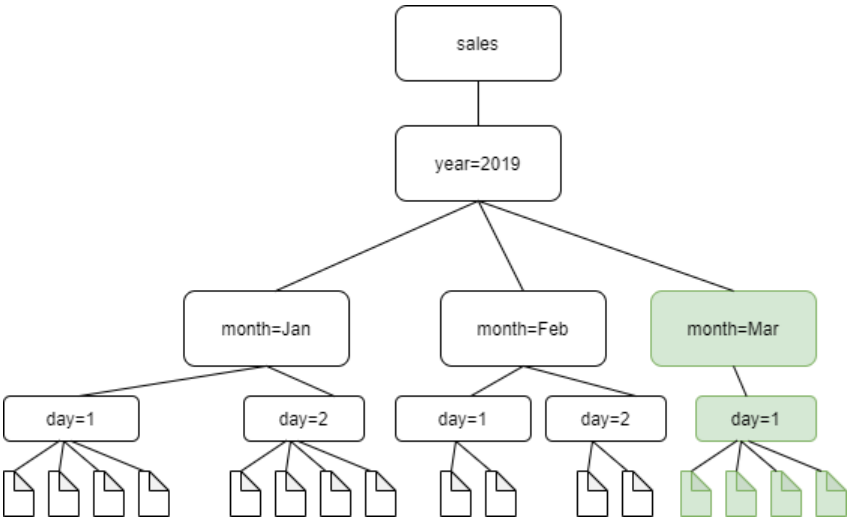
- `LOG`: ignorar el cambio. No actualizar el Catálogo de datos. Escriba un mensaje de registro en su lugar.
- `DEPRECATE_IN_DATABASE`: marcar la tabla como obsoleta en el Catálogo de datos. Este es el valor predeterminado.

Rastreos progresivos para agregar nuevas particiones

El rastreador ofrece una opción para agregar particiones nuevas, lo que resulta en rastreo más rápidos para conjuntos de datos progresivos con un esquema de tabla estable. El caso de uso típico es para rastreadores programados, en los que en cada rastreo se agregan nuevas particiones. Cuando esta opción esté activada, primero ejecutará un rastreo completo en el conjunto de datos de destino para permitir que el rastreador registre el esquema inicial y la estructura de partición. Durante un nuevo rastreo, se agregarán nuevas particiones a tablas existentes solo cuando los esquemas sean compatibles. No se realizan cambios en el esquema ni se agregarán nuevas tablas al Catálogo de datos tras la primera ejecución del rastreo.

Esta opción se puede utilizar al configurar un origen de datos de Amazon S3. Puede configurar la `RecrawlPolicy` con `RecrawlBehavior` como “Crawl_New_Folders” en la API `CreateCrawler` o, Ejecuciones posteriores del rastreador como Rastrear solo nuevas subcarpetas en la consola.

Si retomamos el ejemplo en the section called “¿Cómo determina un rastreador cuándo crear particiones?”, el siguiente diagrama muestra que se han agregado archivos para el mes de marzo.



Si configura `RecrawlBehavior` como “Crawl_New_Folders”, solo se rastreará la nueva carpeta `month=Mar`.

Notas y restricciones

Cuando esta opción está activada, no puede cambiar los almacenes de datos de destino de Amazon S3 al editar el rastreador. Esta opción afecta a determinados valores de configuración del rastreador. Cuando está activada, fuerza el comportamiento de actualización y el comportamiento de eliminación del rastreador a `LOG`. Esto significa que:

- Si descubre objetos en los que los esquemas no son compatibles, el rastreador no agregará los objetos en el Catálogo de datos y agregará este detalle como un registro en CloudWatch Logs.
- No actualizará los objetos eliminados en el Catálogo de datos.

Para obtener más información, consulte the section called “Personalización del comportamiento del rastreador”.

Establecer la opción de configuración del rastreador de índices de particiones

El Data Catalog es compatible con índices de particiones para proporcionar una búsqueda eficiente de particiones específicas. Para obtener más información, consulte Trabajar con índices de partición en AWS Glue. El rastreador de AWS Glue crea, de manera predeterminada, índices de particiones para los objetivos de Amazon S3 y Delta Lake.

Cuando se define un rastreador, la opción para Crear índices de partición de manera automática se habilita de manera predeterminada en la sección Opciones avanzadas en la página Establecer salida y programación.

Para desactivar esta opción, puede anular la selección de la casilla Crear índices de particiones de manera automática en la consola. También puede desactivarla con una API de rastreador al establecer `CreatePartitionIndex` en la `Configuration`. El valor predeterminado es true.

Notas de uso de índices de particiones

- Las tablas creadas por el rastreador no tienen la variable de forma `partition_filtering.enabled` predeterminada. Para obtener más información, consulte Índices de partición y filtros de AWS Glue.
- La creación de índices de partición para particiones cifradas no es compatible.

Aceleración de los rastreadores mediante las notificaciones de eventos de Amazon S3

En lugar de publicar los objetos de un destino de Amazon S3 o del Catálogo de datos, puede configurar el rastreador para que utilice eventos de Amazon S3 para buscar cualquier cambio.

Esta característica mejora el tiempo de rastreo mediante el uso de eventos de Amazon S3 o del Catálogo de datos para identificar los cambios entre dos rastreos al enumerar todos los archivos de la subcarpeta que activó el evento en lugar de publicar el destino completo de Amazon S3 o el Catálogo de datos.

En el primer rastreo se enumeran todos los objetos de Amazon S3 del destino. Después del primer rastreo exitoso, puede optar por volver a rastrear manualmente o según un calendario establecido. El rastreador enumerará solo los objetos de esos eventos en lugar de enumerar todos los objetos.

Las ventajas de pasar a un rastreador basado en eventos de Amazon S3 son:

- Un nuevo rastreo más rápido, ya que la lista de todos los objetos del destino no es necesaria, sino que la lista de carpetas específicas se realiza cuando se agregan o eliminan objetos.
- Reducción del costo global de rastreo a medida que la lista de carpetas específicas se realiza en las que se agregan o eliminan objetos.

El rastreo de eventos de Amazon S3 se ejecuta al consumir eventos de Amazon S3 desde la cola de SQS según la programación del rastreador. No habrá ningún costo si no hay eventos en la cola. Los eventos de Amazon S3 se pueden configurar para que vayan directamente a la cola de SQS o en los casos en que varios consumidores necesitan el mismo evento, una combinación de SNS y SQS. Para obtener más información, consulte the section called “Cómo configurar la cuenta para las notificaciones de eventos de Amazon S3”.

Después de crear y configurar el rastreador en modo evento, el primer rastreo se ejecuta en modo listado y enumera un listado completo del destino de Amazon S3 o del Catálogo de datos. A través del siguiente registro se confirma el funcionamiento del rastreo mediante el uso de eventos de Amazon S3 tras el primer rastreo correcto: “El rastreo se ejecuta mediante el uso de eventos de Amazon S3”.

Después de crear el rastreo de eventos de Amazon S3 y actualizar las propiedades del rastreador que pueden afectar al rastreo, el rastreo funciona en modo lista y se agrega el siguiente registro: “El rastreo no se ejecuta en modo de evento de S3”.

> **Note**
>
> El número máximo de mensajes que se pueden consumir es de 10 000 mensajes por rastreo.

Catálogo de datos como destino

Cuando el destino es el Catálogo de datos, el rastreador actualiza las tablas existentes en el Catálogo de datos con los cambios (por ejemplo, particiones adicionales en una tabla).

Temas

- Cómo configurar la cuenta para las notificaciones de eventos de Amazon S3
- Uso del cifrado con el rastreador de eventos de Amazon S3

Cómo configurar la cuenta para las notificaciones de eventos de Amazon S3

En esta sección, se describe cómo configurar la cuenta para las notificaciones de eventos de Amazon S3 y se proporcionan instrucciones para hacerlo mediante un script o una consola de AWS Glue.

Requisitos previos

Realice los siguientes pasos de configuración. Tenga en cuenta que los valores entre paréntesis hacen referencia a los valores configurables del script.

1. Cree un bucket de Amazon S3 (`s3_bucket_name`)
2. Identificar un objetivo de rastreador (`folder_name`, como “test1") que es una ruta en el bucket identificado.
3. Preparar un nombre de rastreador (`crawler_name`)
4. Prepare un nombre de tema de SNS (`sns_topic_name`) que podría ser el mismo que el nombre del rastreador.
5. Prepare la región de AWS en la que se va a ejecutar el rastreador y donde existe el bucket de S3 (`region`).
6. Si lo desea, prepare una dirección de correo electrónico si se utiliza el correo electrónico para obtener los eventos de Amazon S3 (`subscribing_email`).

También puede usar la pila CloudFormation para crear sus recursos. Realice los siguientes pasos:

1. Lance su pila CloudFormation en el Este de EE. UU. (Norte de Virginia):
2. En Parámetros, ingrese un nombre para su bucket de Amazon S3 (incluye su número de cuenta).
3. Seleccione `I acknowledge that AWS CloudFormation might create IAM resources with custom names`.

4. Elija `Create stack`.

Limitaciones:

- El rastreador admite un solo destino, ya sean destinos para Amazon S3 o para el Catálogo de datos.
- No es posible utilizar SQS en una VPC privada.
- No se admite el muestreo de Amazon S3.
- El destino del rastreador debe ser una carpeta para un destino de Amazon S3 o una o más tablas del Catálogo de datos de AWS Glue para un destino del Catálogo de datos.
- No se admite el comodín de la ruta “todo”: s3://%
- Para un destino de Catálogo de datos, todas las tablas del Catálogo deben apuntar al mismo bucket de Amazon S3 para el modo de eventos de Amazon S3.
- Para un destino de Catálogo de datos, una tabla de catálogo no debe apuntar a una ubicación de Amazon S3 en formato Delta Lake (que contenga carpetas _symlink o consulte las tablas del catálogo `InputFormat`).

Para utilizar el rastreador basado en eventos de Amazon S3, debe habilitar la notificación de eventos en el bucket de S3 con eventos filtrados del prefijo, que es el mismo que el destino de S3 y el almacenamiento en SQS. Puede configurar SQS y la notificación de eventos a través de la consola siguiendo los pasos del Tutorial: configuración de un bucket para notificaciones o mediante the section called “Script para generar SQS y configurar eventos de Amazon S3 desde el destino”.

Política de SQS

Agregue la siguiente política de SQS que debe adjuntarse al rol utilizado por el rastreador.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "sqs:DeleteMessage",
                "sqs:GetQueueUrl",
                "sqs:ListDeadLetterSourceQueues",
```

```
                "sqs:ReceiveMessage",
                "sqs:GetQueueAttributes",
                "sqs:ListQueueTags",
                "sqs:SetQueueAttributes",
                "sqs:PurgeQueue"
            ],
            "Resource": "arn:aws:sqs:{region}:{accountID}:cfn-sqs-queue"
        }
    ]
}
```

Script para generar SQS y configurar eventos de Amazon S3 desde el destino

Una vez garantizado que se cumplen los requisitos previos, puede ejecutar el siguiente script de Python para crear el SQS. Sustituya la configuración configurable por los nombres preparados a partir de los requisitos previos.

> **Note**
>
> Después de ejecutar el script, inicie sesión en la consola SQS para buscar el ARN del SQS creado.

Amazon SQS establece un tiempo de espera de visibilidad, un periodo durante el cual Amazon SQS impide que otros consumidores reciban y procesen el mensaje. Establezca el tiempo de espera de visibilidad aproximadamente igual al tiempo de ejecución de rastreo.

```
#!venv/bin/python
import boto3
import botocore

#---------Start : READ ME FIRST ----------------------#
# 1. Purpose of this script is to create the SQS, SNS and enable S3 bucket
 notification.
#     The following are the operations performed by the scripts:
#      a. Enable S3 bucket notification to trigger 's3:ObjectCreated:' and
 's3:ObjectRemoved:' events.
#      b. Create SNS topic for fan out.
#      c. Create SQS queue for saving events which will be consumed by the crawler.
#          SQS Event Queue ARN will be used to create the crawler after running the
 script.
# 2. This script does not create the crawler.
```

```
# 3. SNS topic is created to support FAN out of S3 events. If S3 event is also used by
 another
#    purpose, SNS topic created by the script can be used.
# 1. Creation of bucket is an optional step.
#    To create a bucket set create_bucket variable to true.
# 2. The purpose of crawler_name is to easily locate the SQS/SNS.
#     crawler_name is used to create SQS and SNS with the same name as crawler.
# 3. 'folder_name' is the target of crawl inside the specified bucket 's3_bucket_name'
#
#---------End : READ ME FIRST ------------------------#


#--------------------------------#
# Start : Configurable settings  #
#--------------------------------#

#Create
region = 'us-west-2'
s3_bucket_name = 's3eventtestuswest2'
folder_name = "test"
crawler_name = "test33S3Event"
sns_topic_name = crawler_name
sqs_queue_name = sns_topic_name
create_bucket = False

#--------------------------------#
# End : Configurable settings   #
#--------------------------------#

# Define aws clients
dev = boto3.session.Session(profile_name='myprofile')
boto3.setup_default_session(profile_name='myprofile')
s3 = boto3.resource('s3', region_name=region)
sns = boto3.client('sns', region_name=region)
sqs = boto3.client('sqs', region_name=region)
client = boto3.client("sts")
account_id = client.get_caller_identity()["Account"]
queue_arn = ""


def print_error(e):
    print(e.message + ' RequestId: ' + e.response['ResponseMetadata']['RequestId'])

def create_s3_bucket(bucket_name, client):
```

```
    bucket = client.Bucket(bucket_name)
    try:
        if not create_bucket:
            return True
        response = bucket.create(
            ACL='private',
            CreateBucketConfiguration={
                'LocationConstraint': region
            },
        )
        return True
    except botocore.exceptions.ClientError as e:
        print_error(e)
        if 'BucketAlreadyOwnedByYou' in e.message:  # we own this bucket so continue
            print('We own the bucket already. Lets continue...')
            return True
    return False

def create_s3_bucket_folder(bucket_name, client, directory_name):
    s3.put_object(Bucket=bucket_name, Key=(directory_name + '/'))

def set_s3_notification_sns(bucket_name, client, topic_arn):
    bucket_notification = client.BucketNotification(bucket_name)
    try:

        response = bucket_notification.put(
            NotificationConfiguration={
                'TopicConfigurations': [
                    {
                        'Id' : crawler_name,
                        'TopicArn': topic_arn,
                        'Events': [
                            's3:ObjectCreated:*',
                            's3:ObjectRemoved:*',

                        ],
                        'Filter' :  {'Key': {'FilterRules': [{'Name': 'prefix',
 'Value': folder_name}]}}
                    },
                ]
            }
        )
        return True
    except botocore.exceptions.ClientError as e:
```

```
        print_error(e)
    return False


def create_sns_topic(topic_name, client):
    try:
        response = client.create_topic(
            Name=topic_name
        )
        return response['TopicArn']
    except botocore.exceptions.ClientError as e:
        print_error(e)
    return None

def set_sns_topic_policy(topic_arn, client, bucket_name):
    try:
        response = client.set_topic_attributes(
            TopicArn=topic_arn,
            AttributeName='Policy',
            AttributeValue='''{
              "Version": "2008-10-17",
              "Id": "s3-publish-to-sns",
              "Statement": [{
                  "Effect": "Allow",
                  "Principal": { "AWS" : "*" },
                  "Action": [ "SNS:Publish" ],
                  "Resource": "%s",
                  "Condition": {
                    "StringEquals": {
                        "AWS:SourceAccount": "%s"
                    },
                    "ArnLike": {
                        "aws:SourceArn": "arn:aws:s3:*:*:%s"
                    }
                }
              }]
  }''' % (topic_arn, account_id, bucket_name)
        )
        return True
    except botocore.exceptions.ClientError as e:
        print_error(e)

    return False
```

```
def subscribe_to_sns_topic(topic_arn, client, protocol, endpoint):
    try:
        response = client.subscribe(
            TopicArn=topic_arn,
            Protocol=protocol,
            Endpoint=endpoint
        )
        return response['SubscriptionArn']
    except botocore.exceptions.ClientError as e:
        print_error(e)
    return None


def create_sqs_queue(queue_name, client):
    try:
        response = client.create_queue(
            QueueName=queue_name,
        )
        return response['QueueUrl']
    except botocore.exceptions.ClientError as e:
        print_error(e)
    return None


def get_sqs_queue_arn(queue_url, client):
    try:
        response = client.get_queue_attributes(
            QueueUrl=queue_url,
            AttributeNames=[
                'QueueArn',
            ]
        )
        return response['Attributes']['QueueArn']
    except botocore.exceptions.ClientError as e:
        print_error(e)
    return None

def set_sqs_policy(queue_url, queue_arn, client, topic_arn):
    try:
        response = client.set_queue_attributes(
            QueueUrl=queue_url,
            Attributes={
                'Policy': '''{
```

```
                  "Version": "2012-10-17",
                  "Id": "AllowSNSPublish",
                  "Statement": [
                    {
                      "Sid": "AllowSNSPublish01",
                      "Effect": "Allow",
                      "Principal": "*",
                      "Action": "SQS:SendMessage",
                      "Resource": "%s",
                      "Condition": {
                        "ArnEquals": {
                          "aws:SourceArn": "%s"
                        }
                      }
                    }
                  ]
}''' % (queue_arn, topic_arn)
            }
        )
        return True
    except botocore.exceptions.ClientError as e:
        print_error(e)
    return False


if __name__ == "__main__":
    print('Creating S3 bucket %s.' % s3_bucket_name)
    if create_s3_bucket(s3_bucket_name, s3):
        print('\nCreating SNS topic %s.' % sns_topic_name)
        topic_arn = create_sns_topic(sns_topic_name, sns)
        if topic_arn:
            print('SNS topic created successfully: %s' % topic_arn)

            print('Creating SQS queue %s' % sqs_queue_name)
            queue_url = create_sqs_queue(sqs_queue_name, sqs)
            if queue_url is not None:
                print('Subscribing sqs queue with sns.')
                queue_arn = get_sqs_queue_arn(queue_url, sqs)
                if queue_arn is not None:
                    if set_sqs_policy(queue_url, queue_arn, sqs, topic_arn):
                        print('Successfully configured queue policy.')
                        subscription_arn = subscribe_to_sns_topic(topic_arn, sns,
 'sqs', queue_arn)
                        if subscription_arn is not None:
```

```
                            if 'pending confirmation' in subscription_arn:
                                print('Please confirm SNS subscription by visiting the
subscribe URL.')
                            else:
                                print('Successfully subscribed SQS queue: ' +
queue_arn)
                        else:
                            print('Failed to subscribe SNS')
                    else:
                        print('Failed to set queue policy.')
                else:
                    print("Failed to get queue arn for %s" % queue_url)
            # ------------ End subscriptions to SNS topic -----------------

            print('\nSetting topic policy to allow s3 bucket %s to publish.' %
s3_bucket_name)
            if set_sns_topic_policy(topic_arn, sns, s3_bucket_name):
                print('SNS topic policy added successfully.')
                if set_s3_notification_sns(s3_bucket_name, s3, topic_arn):
                    print('Successfully configured event for S3 bucket %s' %
s3_bucket_name)
                    print('Create S3 Event Crawler using SQS ARN %s' % queue_arn)
                else:
                    print('Failed to configure S3 bucket notification.')
            else:
                print('Failed to add SNS topic policy.')
        else:
            print('Failed to create SNS topic.')
```

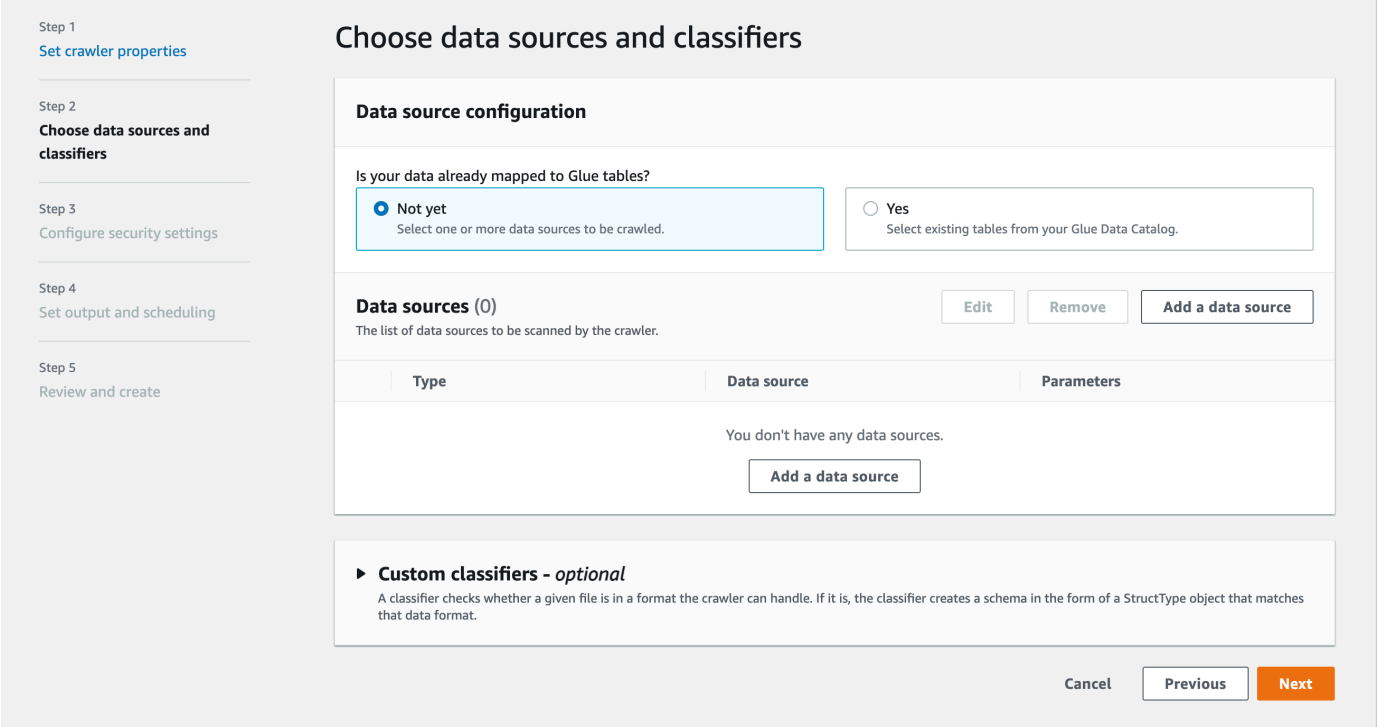
Configuración de un rastreador para notificaciones de eventos de Amazon S3 mediante la consola (Amazon S3 como destino)

Para configurar un rastreador para las notificaciones de eventos de Amazon S3 mediante la consola de AWS Glue para un destino de Amazon S3:

1. Configure las propiedades del rastreador. Para obtener más información, consulte Opciones de configuración de rastreadores en la consola de AWS Glue.
2. En la sección Configuración de origen de datos, se preguntará ¿Los datos ya están asignados a tablas de AWS Glue?

   De manera predeterminada, está seleccionado Not yet (Aún no). Déjelo así si está utilizando un origen de datos de Amazon S3 y los datos aún no están asignados a tablas de AWS Glue.

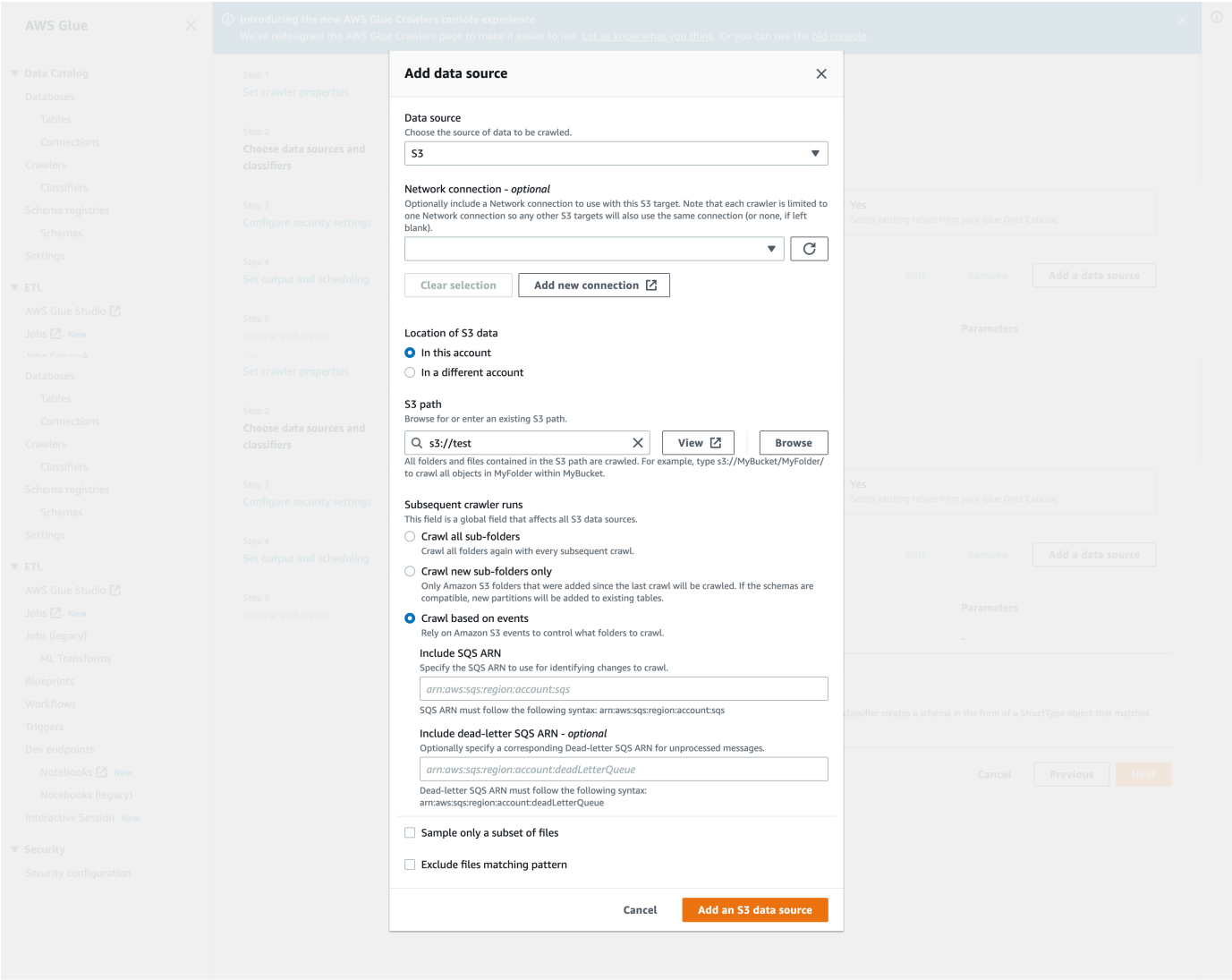
3. En la sección Data sources (Origen de datos), elija Add a data source (Agregar un origen de datos).



4. En el modal Add data source (Agregar origen de datos), configure el origen de datos de Amazon S3:

   - Data source (Origen de datos): de manera predeterminada, está seleccionado Amazon S3.
   - Network connection (Conexión de red) (opcional): elija Add new connection (Agregar nueva conexión).
   - Location of Amazon S3 data (Ubicación de datos de Amazon S3): de manera predeterminada, está seleccionado In this account (En esta cuenta).
   - Amazon S3 path (Ruta de Amazon S3): especifique la ruta de Amazon S3 en la que se rastrean carpetas y archivos.
   - Subsequent crawler runs (Ejecuciones posteriores del rastreador): elija Crawl based on events (Rastreo basado en eventos) para utilizar las notificaciones de eventos de Amazon S3 para el rastreador.
   - Include SQS ARN (Incluir ARN de SQS): especifique los parámetros del almacén de datos, incluido un ARN SQS válido. (Por ejemplo, `arn:aws:sqs:region:account:sqs`).
   - Include dead-letter SQS ARN (Incluir un SQS ARN de mensajes fallidos) (Optional): especifique un ARN de SQS con mensajes erróneos de Amazon válido. (Por ejemplo, `arn:aws:sqs:region:account:deadLetterQueue`).

- Elija Add an Amazon S3 data source (Agregar un origen de datos de Amazon S3).



Configuración de un rastreador para notificaciones de eventos de Amazon S3 mediante la AWS CLI

El siguiente es un ejemplo de llamada a la AWS CLI de Amazon S3 para crear colas de SQS y configurar notificaciones de eventos en el bucket de destino de Amazon S3.

```
S3 Event AWS CLI
aws sqs create-queue --queue-name MyQueue --attributes file://create-queue.json
create-queue.json
```
{

```
    "Policy": {
        "Version": "2012-10-17",
        "Id": "example-ID",
        "Statement": [
            {
                "Sid": "example-statement-ID",
                "Effect": "Allow",
                "Principal": {
                    "Service": "s3.amazonaws.com"
                },
                "Action": [
                    "SQS:SendMessage"
                ],
                "Resource": "SQS-queue-ARN",
                "Condition": {
                    "ArnLike": {
                        "aws:SourceArn": "arn:aws:s3:*:*:awsexamplebucket1"
                    },
                    "StringEquals": {
                        "aws:SourceAccount": "bucket-owner-account-id"
                    }
                }
            }
        ]
    }
}
```

```
aws s3api put-bucket-notification-configuration --bucket customer-data-pdx --
notification-configuration file://s3-event-config.json
s3-event-config.json
```

```
{
    "QueueConfigurations": [
        {
          "Id": "s3event-sqs-queue",
          "QueueArn": "arn:aws:sqs:{region}:{account}:queuename",
          "Events": [
                "s3:ObjectCreated:*",
                "s3:ObjectRemoved:*"
          ],
          "Filter": {
              "Key": {
                 "FilterRules": [
                     {
```

```
                        "Name": "Prefix",
                        "Value": "/json"
                    }
                ]
            }
        }
    ]
}
```

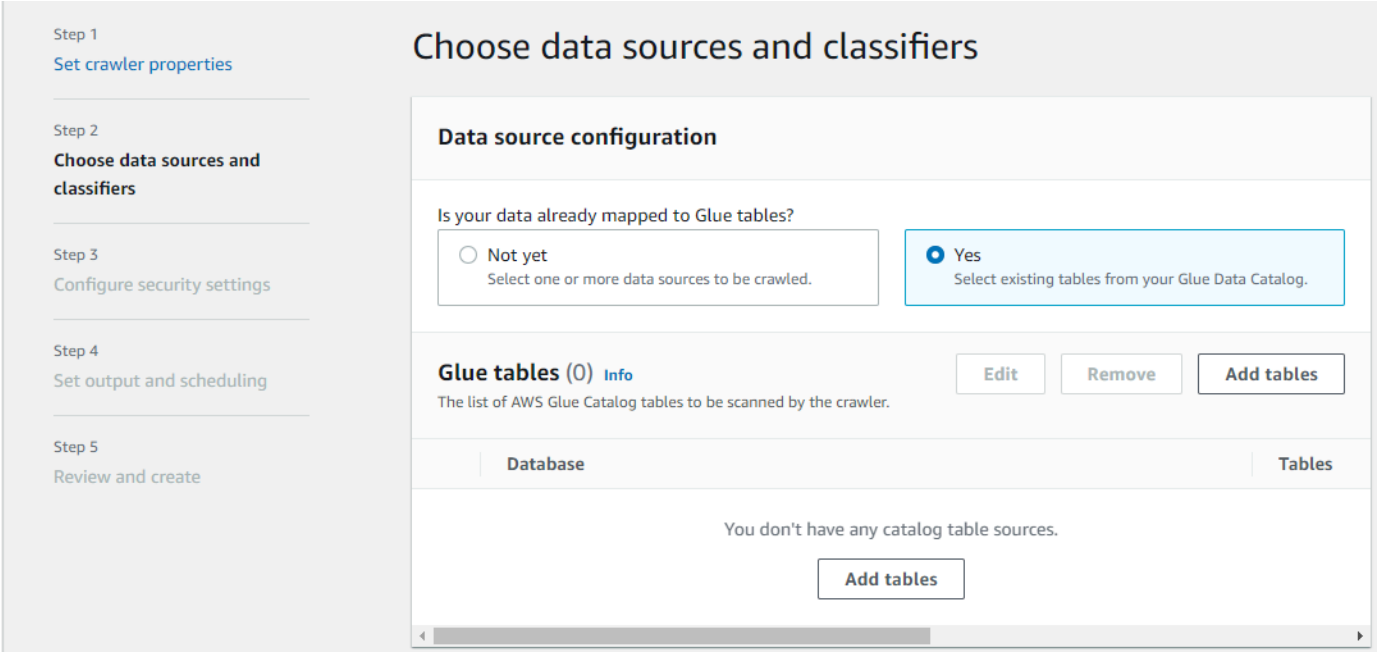
Create Crawler:

Configuración de un rastreador para notificaciones de eventos de Amazon S3 mediante la consola (el Catálogo de datos como destino)

Cuando el destino sea un catálogo, configure un rastreador para las notificaciones de eventos de Amazon S3 mediante la consola de AWS Glue:

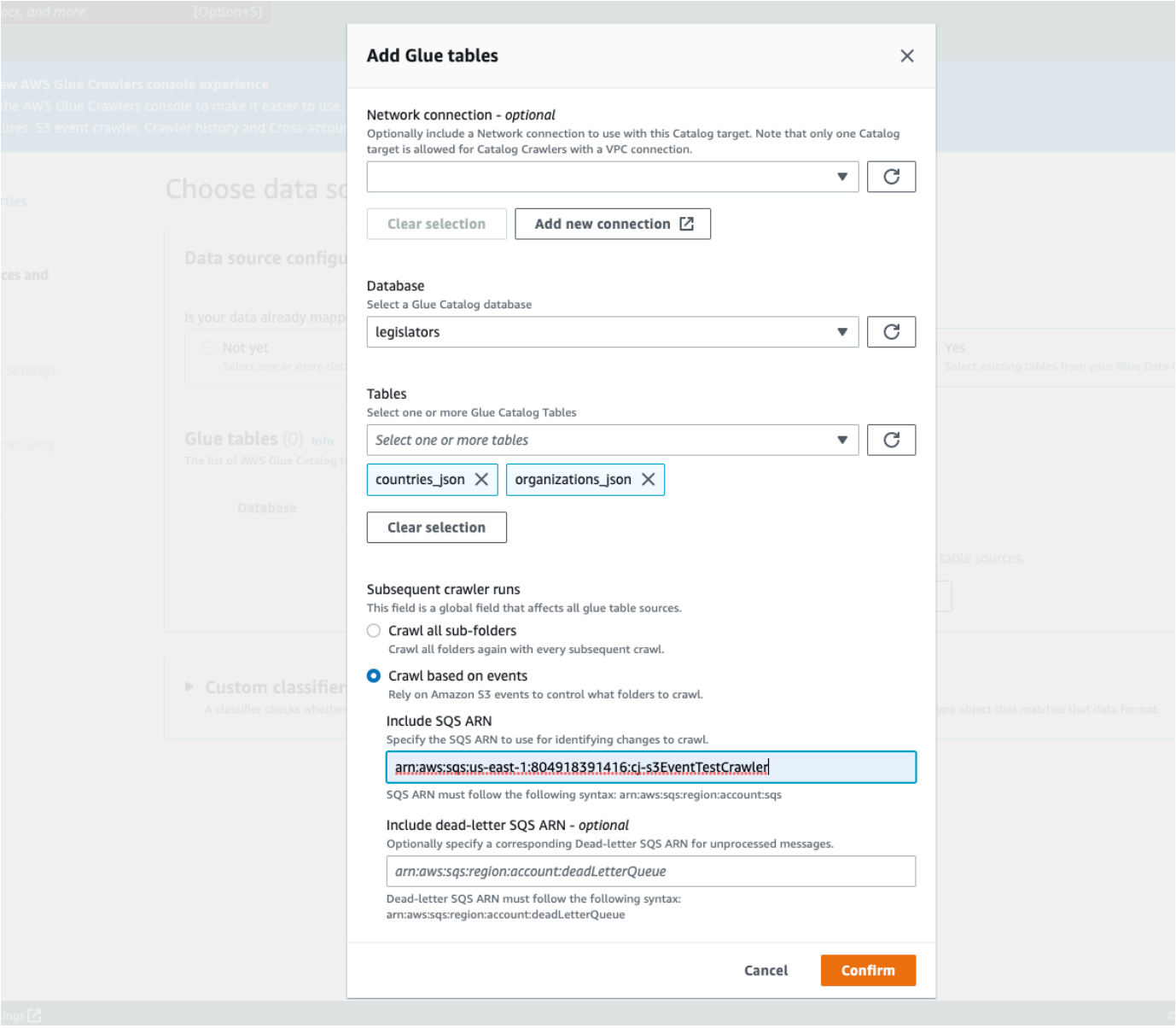
1. Configure las propiedades del rastreador. Para obtener más información, consulte Opciones de configuración de rastreadores en la consola de AWS Glue.
2. En la sección Configuración de origen de datos, se preguntará ¿Los datos ya están asignados a tablas de AWS Glue?

   Seleccione Yes (Sí) para seleccionar las tablas existentes de su Catálogo de datos como origen de datos.
3. En la sección Glue tables (Tablas de Glue), seleccione Add tables (Agregar tablas).



4. En el modal Add table (Agregar tabla), configure la base de datos y las tablas:

   - Network connection (Conexión de red) (opcional): elija Add new connection (Agregar nueva conexión).
   - Base de datos: seleccione una base de datos en el Catálogo de datos.
   - Tablas: seleccione una o más tablas de esa base de datos en el Catálogo de datos.
   - Subsequent crawler runs (Ejecuciones posteriores del rastreador): elija Crawl based on events (Rastreo basado en eventos) para utilizar las notificaciones de eventos de Amazon S3 para el rastreador.
   - Include SQS ARN (Incluir ARN de SQS): especifique los parámetros del almacén de datos, incluido un ARN SQS válido. (Por ejemplo, `arn:aws:sqs:region:account:sqs`).
   - Include dead-letter SQS ARN (Incluir un SQS ARN de mensajes fallidos) (Optional): especifique un ARN de SQS con mensajes erróneos de Amazon válido. (Por ejemplo, `arn:aws:sqs:region:account:deadLetterQueue`).
   - Elija Confirmar.



Uso del cifrado con el rastreador de eventos de Amazon S3

En esta sección, se describe el uso del cifrado solo en SQS o en SQS y Amazon S3.

Temas

- Habilitación del cifrado solo en SQS
- Activación del cifrado en SQS y Amazon S3
- Preguntas frecuentes

Habilitación del cifrado solo en SQS

Amazon SQS proporciona cifrado en tránsito de forma predeterminada. Para agregar el cifrado del lado del servidor (SSE) opcional a la cola, puede adjuntar una clave maestra del cliente (CMK) en el panel de edición. Esto significa que SQS cifra todos los datos de los clientes en reposo en los servidores SQS.

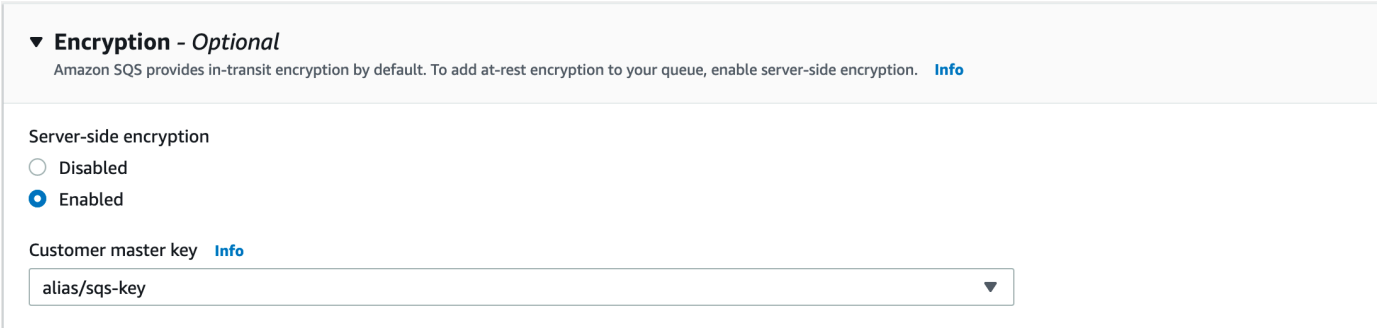
Crear una clave maestra del cliente (CMK)

1. Elija Key Management Service (KMS) > Customer Managed Key (Claves administradas por el cliente) > Create key (Creación de claves).
2. Siga los pasos para agregar su propio alias y descripción.
3. Agregue los respectivos roles de IAM que le gustaría que puedan utilizar esta clave.
4. En la política clave, agregue otra declaración a la lista “Declaración” para que la política de claves personalizada otorga a Amazon SNS permisos de uso de claves suficientes.

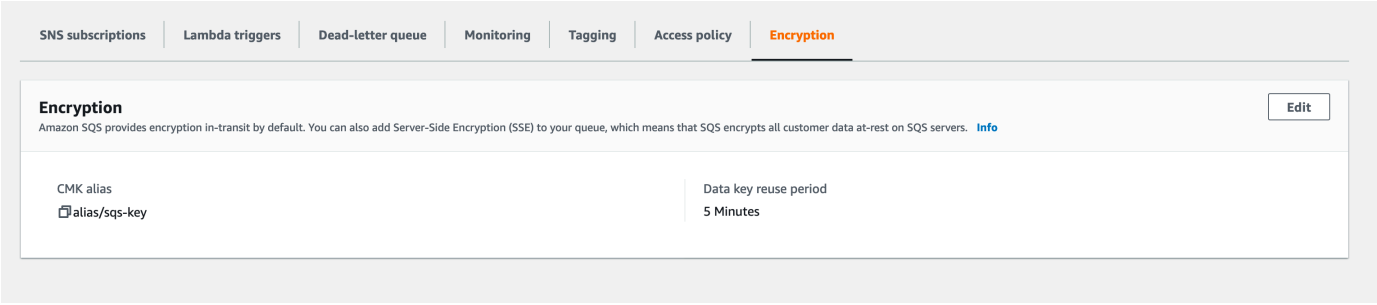
```
"Statement": [
       {
           "Effect": "Allow",
           "Principal": {
               "Service": "sns.amazonaws.com"
           },
           "Action": [
               "kms:GenerateDataKey",
               "kms:Decrypt"
           ],
           "Resource": "*"
       }
]
```

Habilitación del cifrado del lado del servidor (SSE) en la cola

1. Elija Amazon SQS >Queues (Colas) > sqs_queue_name > pestaña Encryption (Cifrado).
2. Elija Edit (Editar) y vaya a Encryption (Cifrado) del menú desplegable.
3. Seleccione Enabled (Habilitado) para agregar SSE.
4. Seleccione la CMK que creó con anterioridad y no la clave predeterminada con el nombre `alias/aws/sqs`.



Después de agregarlo, la pestaña Encryption (Cifrado) se actualiza con la clave que ha agregado.



> **Note**
>
> Amazon SQS elimina automáticamente los mensajes que han estado en una cola durante más tiempo que el periodo máximo de retención de mensajes. El periodo de retención de mensajes predeterminado es de 4 días. Para evitar que falten eventos, cambie el SQS MessageRetentionPeriod hasta un máximo de 14 días.

Activación del cifrado en SQS y Amazon S3

Habilitación del cifrado del lado del servidor (SSE) en SQS

1. Siga los pasos de the section called “Habilitación del cifrado solo en SQS”.
2. En el último paso de la configuración de CMK, otorgue a Amazon S3 permisos de uso de claves suficientes.

   Pegue lo siguiente en la lista “Declaración”:

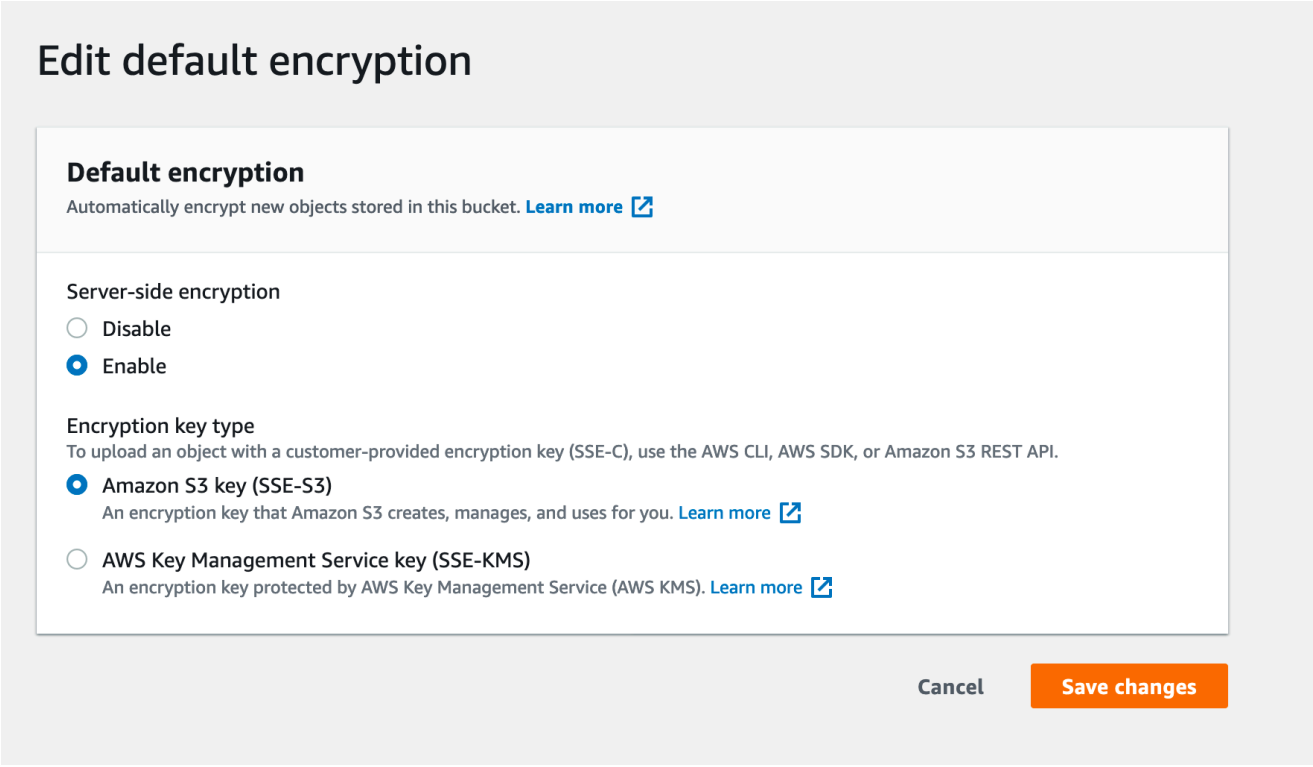
```
"Statement": [
      {
```

```
            "Effect": "Allow",
            "Principal": {
                "Service": "s3.amazonaws.com"
            },
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": "*"
        }
]
```

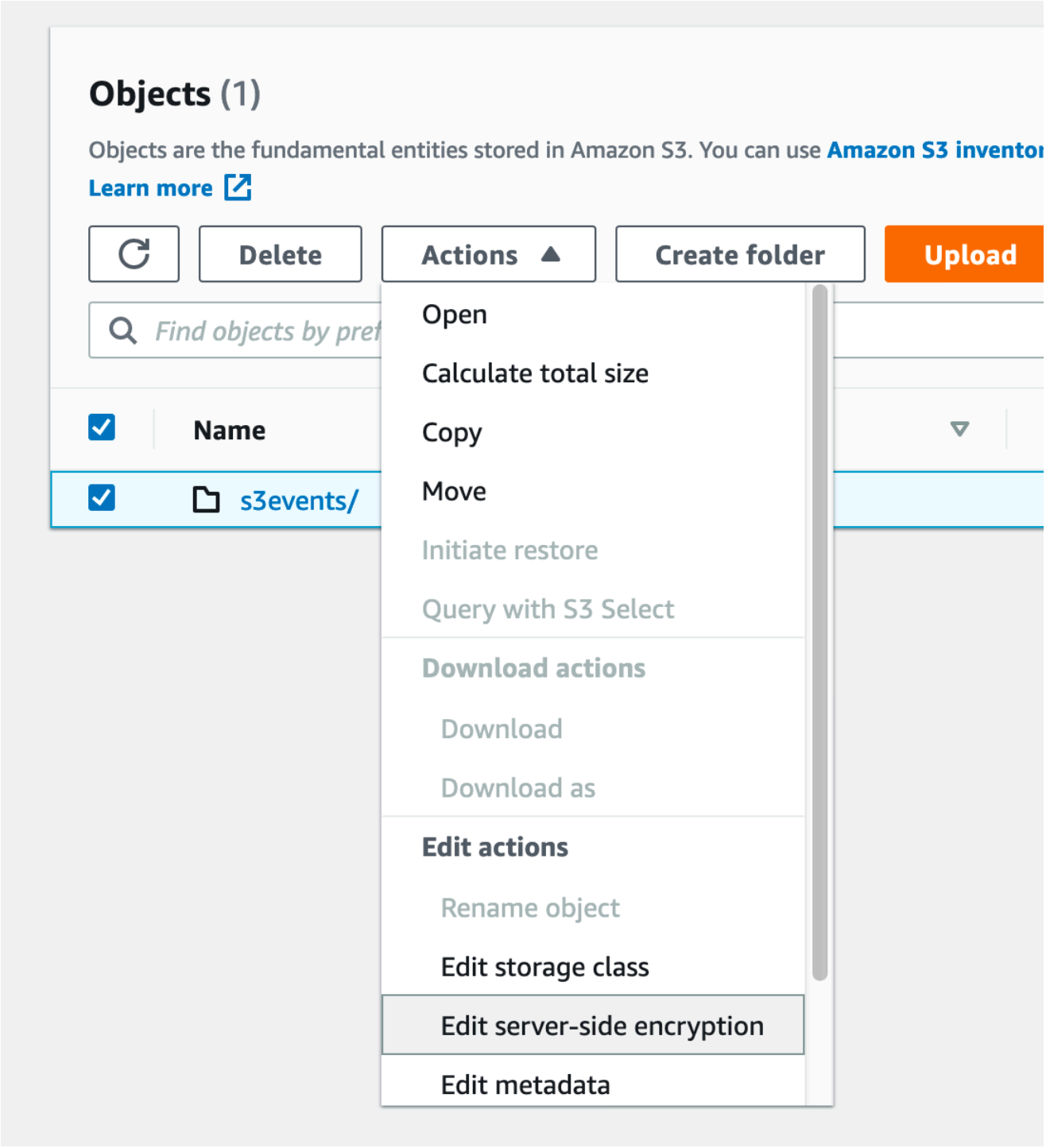
Habilitación del cifrado del servidor (SSE) en el bucket de Amazon S3

1. Siga los pasos de the section called “Habilitación del cifrado solo en SQS”.
2. Realice una de las acciones siguientes:
   - Para habilitar SSE para todo el bucket de S3, navegue hasta la pestaña Properties (Propiedades) en el bucket de destino.

     Aquí puede habilitar SSE y elegir el tipo de cifrado que le gustaría utilizar. Amazon S3 proporciona una clave de cifrado que Amazon S3 crea, administra y utiliza por usted, o también puede elegir una clave de KMS.



- Para habilitar SSE en una carpeta específica, haga clic en la casilla de verificación situada junto a la carpeta de destino y elija Edit server-side encryption (Edición del cifrado del lado del servidor) en el menú desplegable Actions (Acciones).



Preguntas frecuentes

¿Por qué los mensajes que publico en mi tema de Amazon SNS no se entregan a mi cola de Amazon SQS suscrita que tiene habilitado el cifrado del lado del servidor (SSE)?

Verifique que la cola de Amazon SQS utiliza:

1. Una clave maestra del cliente (CMK) que es administrada por el cliente. No es la clave predeterminada proporcionada por SQS.
2. Su CMK de (1) incluye una política de claves personalizada que otorga a Amazon SNS permisos de uso de claves suficientes.

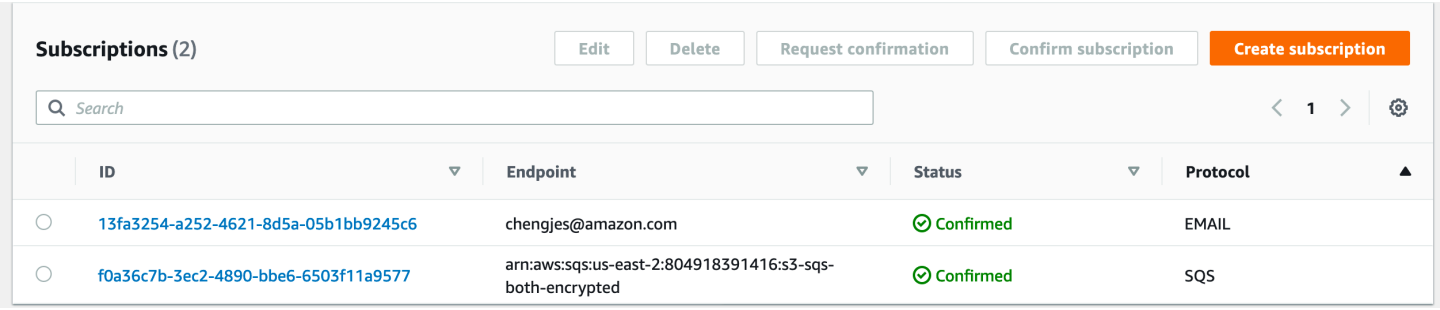
Para obtener más información, consulte este artículo en el centro de conocimientos.

Me he suscrito a notificaciones por correo electrónico, pero no recibo ninguna actualización por correo electrónico cuando edito mi bucket de Amazon S3.

Asegúrese de haber confirmado su dirección de correo electrónico al hacer clic en el enlace "Confirm Subscription" (Confirmar suscripción) del correo electrónico. Puede verificar el estado de la confirmación mediante la verificación de la tabla Subscriptions (Suscripciones) en el tema de SNS.

Elija Amazon SNS >Topics (Temas) > `sns_topic_name` > Subscriptions table (Tabla de suscripciones).

Si siguió nuestro script de requisitos previos, descubrirá que el `sns_topic_name` es igual que `sqs_queue_name`. Debería parecerse a lo que sigue:



Solo algunas de las carpetas que agregué aparecen en mi tabla después de habilitar el cifrado del lado del servidor en mi cola SQS. ¿Por qué me faltan algunos parquets?

Si los cambios del bucket de Amazon S3 se realizaron antes de habilitar SSE en la cola de SQS, es posible que el rastreador no los recoja. Para asegurarse de que ha rastreado todas las actualizaciones del bucket de S3, vuelva a ejecutar el rastreador en modo listado ("Rastrear todas las carpetas"). Otra opción es iniciar de nuevo y crear un nuevo rastreador con eventos de S3 habilitados.

Cómo evitar que el rastreador cambie un esquema existente

Si no desea que un rastreador sobrescriba las actualizaciones que haya realizado en los campos existentes de una definición de tabla de Amazon S3, elija la opción en la consola Agregar solo

columnas nuevas o establezca la opción de configuración `MergeNewColumns`. Esto se aplica a las tablas y las particiones, a menos que `Partitions.AddOrUpdateBehavior` se sustituya por `InheritFromTable`.

Si no desea que un esquema de tabla cambie en absoluto cuando se ejecuta un rastreador, establezca la política de cambio de esquema en `LOG`. También puede definir una opción de configuración que establezca los esquemas de partición que se heredarán de la tabla.

Si está configurando el rastreador en la consola, puede elegir las siguientes acciones:

- Ignorar el cambio y no actualizar la tabla en el Catálogo de datos
- Actualizar todas las particiones nuevas y existentes con metadatos de la tabla

Al configurar el rastreador mediante la API, establezca los siguientes parámetros:

- Configure el campo `UpdateBehavior` de la estructura `SchemaChangePolicy` en `LOG`.
- Establezca el campo `Configuration` con una representación de cadena del siguiente objeto JSON en la API del rastreador; por ejemplo:

```
{
   "Version": 1.0,
   "CrawlerOutput": {
      "Partitions": { "AddOrUpdateBehavior": "InheritFromTable" }
   }
}
```

Cómo crear un único esquema para cada ruta de inclusión de Amazon S3

De forma predeterminada, cuando un rastreador define tablas para los datos almacenados en Amazon S3, tiene en cuenta la compatibilidad y el esquema de los datos por igual. Entre los factores de compatibilidad de datos que se tienen en cuenta se incluye si los datos poseen el mismo formato (por ejemplo, JSON), el mismo tipo de compresión (por ejemplo, GZIP), la estructura de la ruta de Amazon S3 y otros atributos de datos. La similitud de los esquemas es una medida de qué tan similares son los esquemas de objetos de Amazon S3 independientes.

Puede configurar un rastreador en `CombineCompatibleSchemas` en una definición de tabla común cuando sea posible. Con esta opción, el rastreador sigue teniendo en cuenta la compatibilidad de los

datos, pero pasa por alto la similitud de los esquemas específicos al evaluar objetos de Amazon S3 en la ruta de inclusión especificada.

Si configura el rastreador en la consola, para combinar los esquemas, seleccione la opción del rastreador Crear un solo esquema para cada ruta de S3.

Al configurar el rastreador mediante la API, establezca la siguiente opción de configuración:

- Establezca el campo `Configuration` con una representación de cadena del siguiente objeto JSON en la API del rastreador; por ejemplo:

```
{
   "Version": 1.0,
   "Grouping": {
      "TableGroupingPolicy": "CombineCompatibleSchemas" }
}
```

Para ayudar a ilustrar esta opción, supongamos que define un rastreador con una ruta de inclusión `s3://bucket/table1/`. Al ejecutarse el rastreador, encuentra dos archivos JSON con las siguientes características:

- Archivo 1 `S3://bucket/table1/year=2017/data1.json`
- Contenido del archivo: `{“A”: 1, “B”: 2}`
- Esquema: `A:int, B:int`

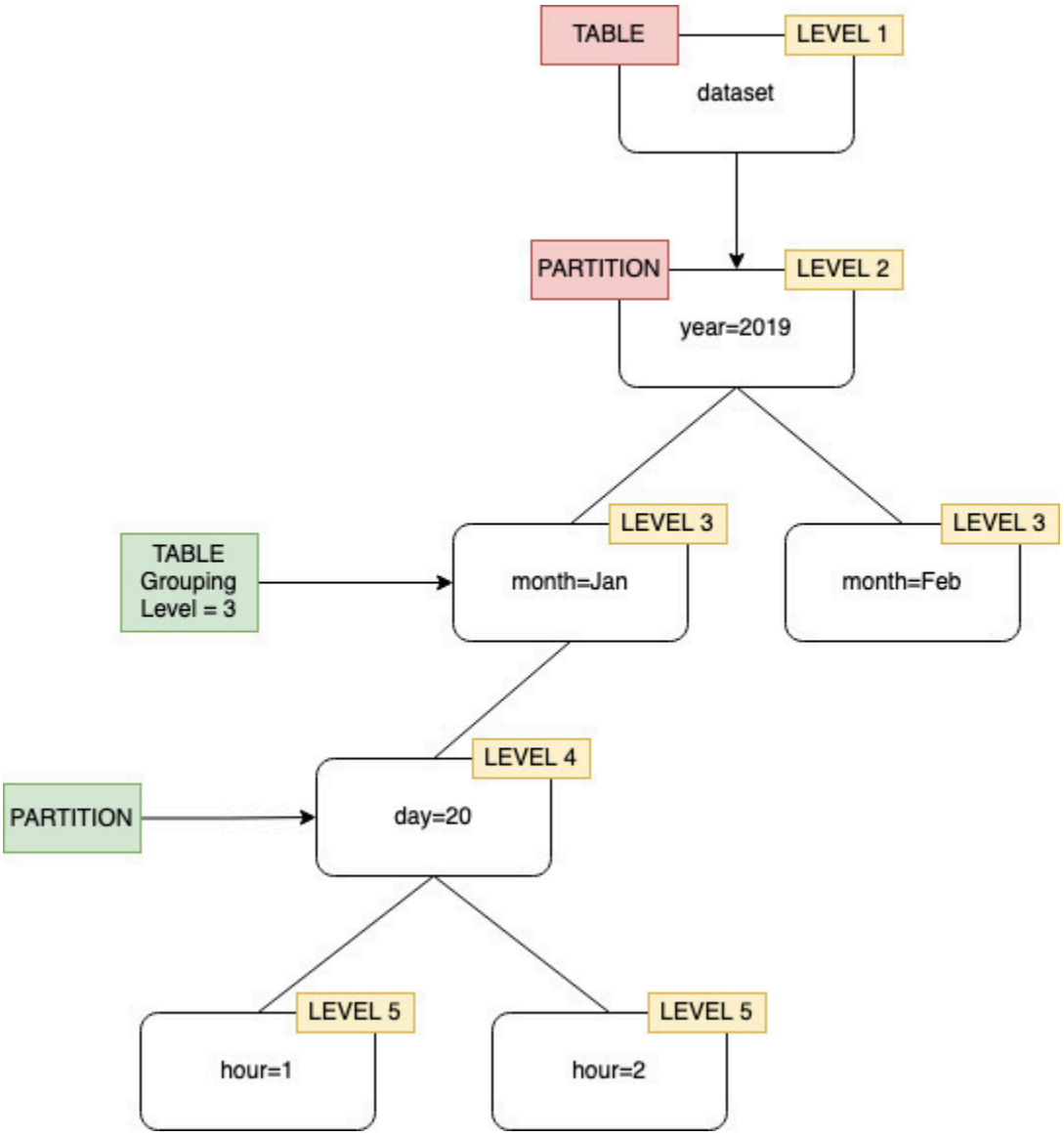
- Archivo 2 `S3://bucket/table1/year=2018/data2.json`
- Contenido del archivo: `{“C”: 3, “D”: 4}`
- Esquema: `C: int, D: int`

De forma predeterminada, el rastreador crea dos tablas, llamadas `year_2017` y `year_2018`, ya que los esquemas no son lo suficientemente similares. Sin embargo, si la opción Crear un solo esquema para cada ruta de S3 está seleccionada y los datos son compatibles, el rastreador crea una tabla. La tabla tiene el esquema `A:int,B:int,C:int,D:int` y `partitionKey year:string`.

Cómo especificar la ubicación de la tabla y el nivel de partición

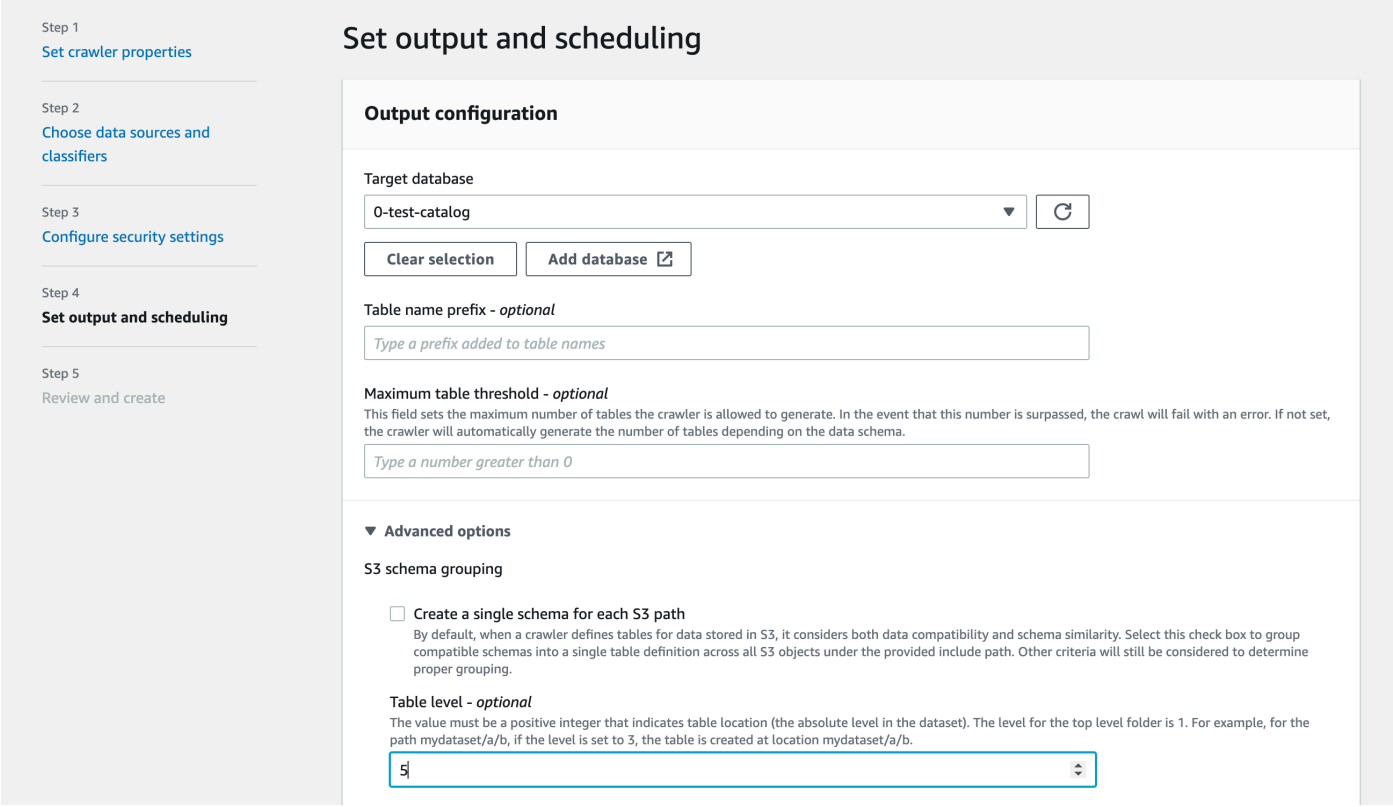
De forma predeterminada, cuando un rastreador define tablas para los datos almacenados en Amazon S3, el rastreador intenta combinar esquemas y crear tablas de nivel superior (`year=2019`). En algunos casos, es posible que espere que el rastreador cree una tabla para la carpeta `month=Jan`, pero en su lugar el rastreador crea una partición ya que una carpeta "hermana" (`month=Mar`) se fusionó en la misma tabla.

La opción de rastreador de nivel de tabla proporciona la flexibilidad para indicar al rastreador dónde se encuentran las tablas y cómo desea crear las particiones. Cuando se especifica un Nivel de tabla, la tabla se crea en ese nivel absoluto a partir del bucket de Amazon S3.



Al configurar el rastreador en la consola, puede especificar un valor para la opción del rastreador Nivel de tabla. El valor debe ser un entero positivo que indique la ubicación de la tabla (el nivel absoluto del conjunto de datos). El nivel de la carpeta de nivel superior es 1. Por ejemplo, para la ruta `mydataset/year/month/day/hour`, si el nivel se establece en 3, la tabla se crea en la ubicación `mydataset/year/month`.

Console



API

Cuando configure el rastreador mediante la API, establezca el campo `Configuration` con una representación de cadena del siguiente objeto JSON; por ejemplo:

```
configuration = jsonencode(

{
   "Version": 1.0,
   "Grouping": {
            TableLevelConfiguration = 2
        }
})
```

CloudFormation

En este ejemplo, establezca la opción Nivel de la tabla como disponible en la consola de la plantilla de CloudFormation:

```
"Configuration": "{
    \"Version\":1.0,
    \"Grouping\":{\"TableLevelConfiguration\":2}
}"
```

Cómo especificar el número máximo de tablas que el rastreador tiene permitido crear

Opcionalmente, puede especificar el número máximo de tablas que el rastreador tiene permitido crear especificando un valor de `TableThreshold` mediante la consola o la CLI de AWS Glue. Si las tablas detectadas por el rastreador durante el rastreo superan este valor de entrada, se produce un error en el rastreo y no se escribe ningún dato en el Catálogo de datos.

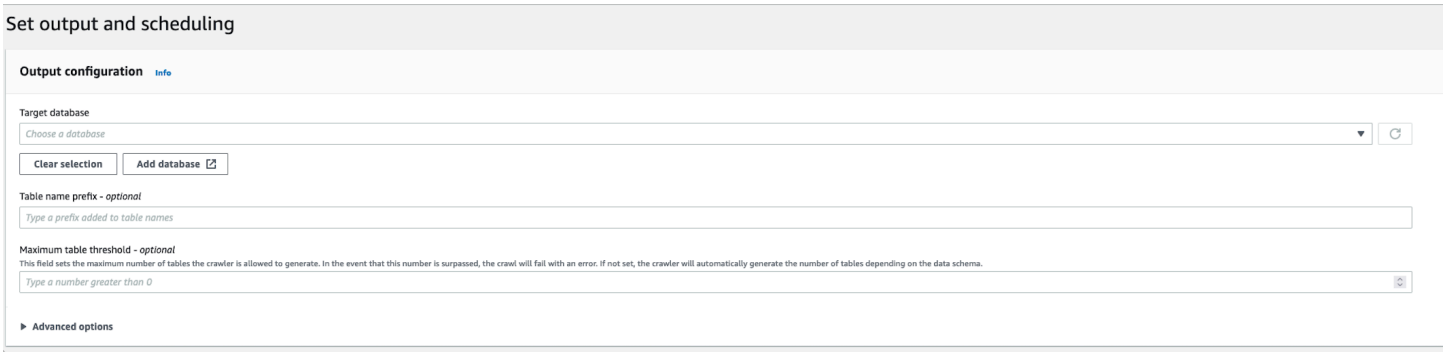
Este parámetro es útil cuando las tablas que detectaría y crearía el rastreador son muchas más de las esperadas. Las razones para que ocurra esto pueden ser varias; por ejemplo:

- Cuando se utiliza un trabajo de AWS Glue para rellenar las ubicaciones de Amazon S3, es posible que acabe habiendo archivos vacíos en el mismo nivel que el de una carpeta. En esos casos, cuando se ejecuta un rastreador en esta ubicación de Amazon S3, el rastreador crea varias tablas debido a la presencia de archivos y carpetas en el mismo nivel.
- Si no configura `"TableGroupingPolicy": "CombineCompatibleSchemas"`, es posible que acabe habiendo más tablas de las esperadas.

Se debe especificar un valor entero mayor que 0 para `TableThreshold`. Este valor se configura para cada rastreador. Es decir, se tiene en cuenta este valor para cada rastreo. Por ejemplo, un rastreador tiene el valor `TableThreshold` establecido en 5. En cada rastreo, AWS Glue compara el número de tablas detectadas con este valor de umbral de tablas (5) y, si el número de tablas detectadas es inferior a 5, AWS Glue escribe las tablas en el Catálogo de datos; en caso contrario, se produce un error en el rastreo y no se escribe nada en el Catálogo de datos.

Consola

Para establecer `TableThreshold` mediante la consola de AWS



CLI

Para configurar `TableThreshold` mediante la CLI de AWS:

```
"{"Version":1.0,
"CrawlerOutput":
{"Tables":{"AddOrUpdateBehavior":"MergeNewColumns",
"TableThreshold":5}}}";
```

Se registran mensajes de error para ayudarle a identificar las rutas de las tablas y a limpiar los datos. Ejemplo de registro en la cuenta si el rastreador falla porque el número de tablas es mayor que el valor umbral de tablas proporcionado:

```
Table Threshold value = 28, Tables detected - 29
```

En CloudWatch, se registran todas las ubicaciones de las tablas detectadas en forma de mensaje INFO. Se registra un error como motivo del fallo.

```
ERROR com.amazonaws.services.glue.customerLogs.CustomerLogService - CustomerLogService
 received CustomerFacingException with message
The number of tables detected by crawler: 29 is greater than the table threshold value
 provided: 28. Failing crawler without writing to Data Catalog.
com.amazonaws.services.glue.exceptions.CustomerFacingInternalException: The number of
 tables detected by crawler: 29 is greater than the table threshold value provided:
 28.
Failing crawler without writing to Data Catalog.
```

Cómo especificar opciones de configuración para un almacén de datos de Delta Lake

Al configurar un rastreador para un almacén de datos de Delta Lake, debe especificar estos parámetros de configuración:

Connection

Si lo desea, seleccione o agregue una conexión de red para utilizarla con este destino de Simple Storage Service (Amazon S3). Para obtener más información acerca de las conexiones, consulte Conexión a datos.

Creación de tablas para realizar consultas

Seleccione cómo desea crear las tablas de Delta Lake:

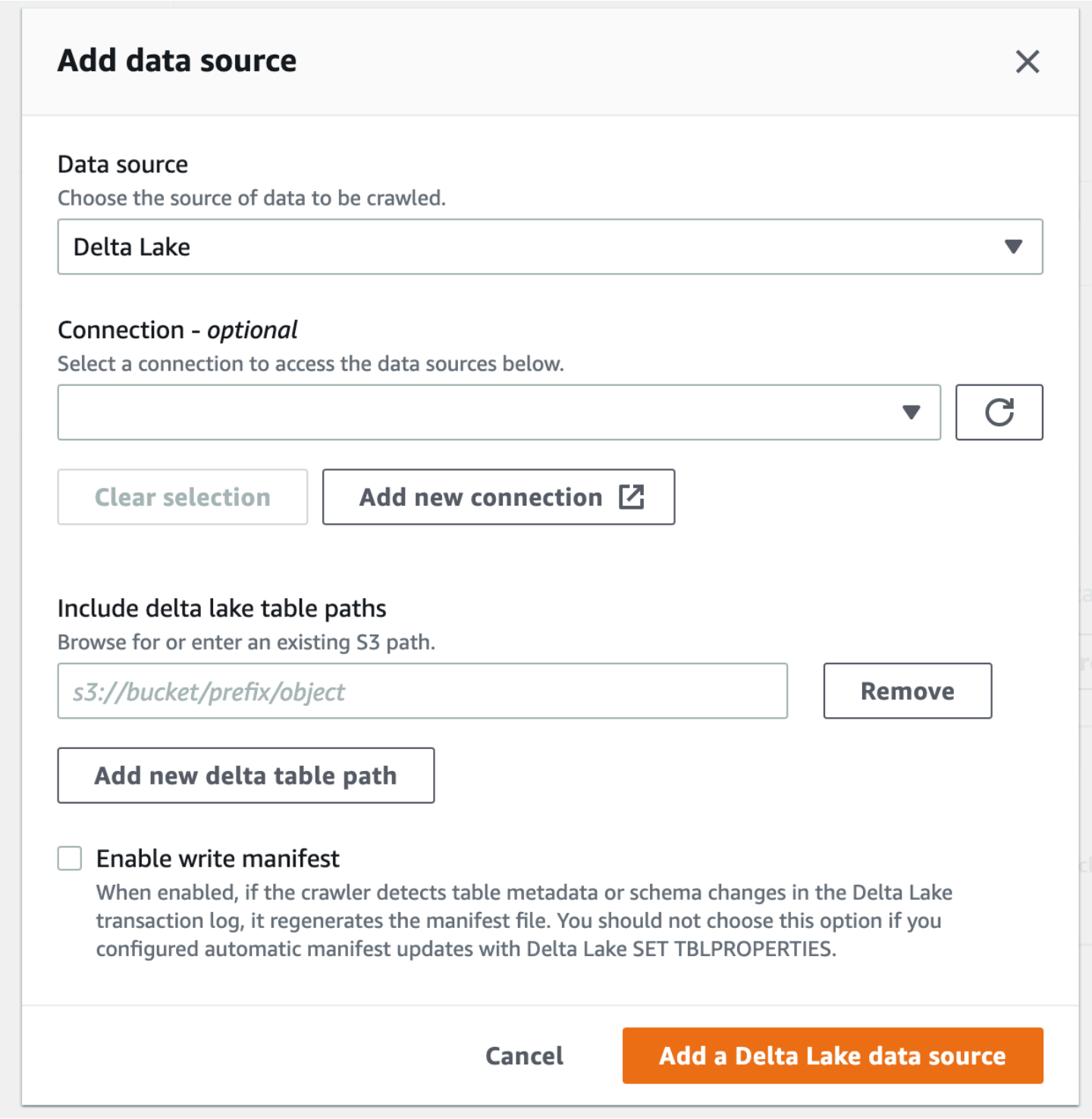
- Crear tablas nativas: se permite la integración con los motores de consulta que permiten consultar directamente el registro de transacciones de Delta.
- Crear tablas de enlaces simbólicos: se crea una carpeta de manifiesto de enlaces simbólicos con los archivos de manifiesto particionados mediante las claves de partición en función de los parámetros de configuración especificados.

Habilite el manifiesto de escritura (configurable solo si ha seleccionado Crear tablas de enlaces simbólicos para una fuente de Delta Lake).

Seleccione si desea detectar metadatos de tablas o cambios de esquemas en el registro de transacciones de Delta Lake; regenera el archivo de manifiesto. No debe elegir esta opción si ha configurado una actualización automática de manifiesto con `SET TBLPROPERTIES` de Delta Lake.

Incluir rutas de tablas de Delta Lake

Especifique una o más rutas de Simple Storage Service (Amazon S3) a las tablas Delta como s3://*`bucket`*/*`prefijo`*/*`objeto`*.



Cómo configurar un rastreador para que utilice credenciales de Lake Formation

Puede configurar un rastreador para que utilice credenciales de AWS Lake Formation para acceder a un almacén de datos de Amazon S3 o a una tabla del Catálogo de datos con una ubicación de Amazon S3 subyacente en la misma Cuenta de AWS o en otra Cuenta de AWS. Puede configurar

una tabla existente del Catálogo de datos como destino de un rastreador, si el rastreador y la tabla del Catálogo de datos residen en la misma cuenta. Actualmente, solo se permite un único destino de catálogo con una sola tabla de catálogo cuando se utiliza una tabla del Catálogo de datos como destino de un rastreador.

> **Note**
>
> Cuando vaya a definir una tabla del Catálogo de datos como destino de un rastreador, asegúrese de que la ubicación subyacente de la tabla del Catálogo de datos sea una ubicación de Amazon S3. Los rastreadores que utilizan credenciales de Lake Formation solo admiten destinos del Catálogo de datos con ubicaciones de Amazon S3 subyacentes.

Configuración requerida cuando el rastreador y la ubicación de Amazon S3 o la tabla del Catálogo de datos registradas residen en la misma cuenta (rastreo en cuenta)

Para permitir que el rastreador acceda a un almacén de datos o a una tabla del Catálogo de datos con credenciales de Lake Formation, se debe registrar la ubicación de los datos en Lake Formation. Además, el rol de IAM del rastreador debe tener permisos para leer los datos del destino en el que esté registrado el bucket de Amazon S3.

Puede completar los siguientes pasos de configuración mediante la AWS Management Console o la AWS Command Line Interface (AWS CLI).

AWS Management Console

1. Antes de configurar un rastreador para que acceda al origen del rastreador, registre la ubicación de los datos del almacén de datos o el Catálogo de datos en Lake Formation. En la consola de Lake Formation (https://console.aws.amazon.com/lakeformation/), registre una ubicación de Amazon S3 como ubicación raíz del lago de datos en la Cuenta de AWS donde esté definido el rastreador. Para obtener más información, consulte Registering an Amazon S3 location (Registro de una ubicación de Amazon S3).
2. Conceda permisos de Ubicación de datos al rol de IAM que se utiliza para la ejecución del rastreador, de modo que el rastreador pueda leer los datos del destino en Lake Formation. Para obtener más información, consulte Concesión de permisos de ubicación de datos (misma cuenta).
3. Otorgue al rol del rastreador permisos de acceso (`Create`) a la base de datos, que se especifica como base de datos de salida. Para obtener más información, consulte Concesión

de permisos de base de datos mediante la consola de Lake Formation y el método de recurso con nombre.

4. En la consola de IAM (https://console.aws.amazon.com/iam/), cree un rol de IAM para el rastreador. Agregue la política `lakeformation:GetDataAccess` al rol.
5. En la consola de AWS Glue (https://console.aws.amazon.com/glue/), al configurar el rastreador, seleccione la opción Utilizar credenciales de Lake Formation para rastrear un origen de datos de Amazon S3.

> **Note**
>
> El campo accountId es opcional para el rastreo en cuenta.

AWS CLI

```
aws glue --profile demo create-crawler --debug --cli-input-json '{
    "Name": "prod-test-crawler",
    "Role": "arn:aws:iam::111122223333:role/service-role/AWSGlueServiceRole-prod-
test-run-role",
    "DatabaseName": "prod-run-db",
    "Description": "",
    "Targets": {
    "S3Targets":[
                {
                 "Path": "s3://crawl-testbucket"
                }
                ]
                },
   "SchemaChangePolicy": {
      "UpdateBehavior": "LOG",
      "DeleteBehavior": "LOG"
  },
  "RecrawlPolicy": {
    "RecrawlBehavior": "CRAWL_EVERYTHING"
  },
  "LineageConfiguration": {
    "CrawlerLineageSettings": "DISABLE"
  },
  "LakeFormationConfiguration": {
    "UseLakeFormationCredentials": true,
    "AccountId": "111122223333"
```

```
  },
  "Configuration": {
           "Version": 1.0,
           "CrawlerOutput": {
             "Partitions": { "AddOrUpdateBehavior": "InheritFromTable" },
             "Tables": {"AddOrUpdateBehavior": "MergeNewColumns" }
           },
           "Grouping": { "TableGroupingPolicy": "CombineCompatibleSchemas" }
         },
  "CrawlerSecurityConfiguration": "",
  "Tags": {
    "KeyName": ""
  }
}'
```

Configuración requerida cuando el rastreador y la ubicación de Amazon S3 registrada residen en cuentas diferentes (rastreo entre cuentas)

Para permitir que el rastreador acceda a un almacén de datos de una cuenta diferente con credenciales de Lake Formation, primero debe registrar la ubicación de los datos de Amazon S3 en Lake Formation. Después, debe conceder permisos de ubicación de datos a la cuenta del rastreador siguiendo estos pasos.

Puede completar los siguientes pasos mediante la AWS Management Console o la AWS CLI.

AWS Management Console

1. En la cuenta en la que esté registrada la ubicación de Amazon S3 (cuenta B):
   a. Registre una ruta de Amazon S3 en Lake Formation. Para obtener más información, consulte Registro de una ubicación de Amazon S3.
   b. Conceda permisos de Ubicación de datos a la cuenta (cuenta A) en la que se vaya a ejecutar el rastreador. Para obtener más información, consulte Concesión de permisos de ubicación de datos.
   c. Cree una base de datos vacía en Lake Formation con la ubicación subyacente como ubicación de Amazon S3 de destino. Para obtener más información, consulte Creación de una base de datos.
   d. Conceda a la cuenta A (la cuenta en la que se vaya a ejecutar el rastreador) acceso a la base de datos que ha creado en el paso anterior. Para obtener más información, consulte Concesión de permisos de base de datos.

2. En la cuenta donde se ha creado y se va a ejecutar el rastreador (cuenta A):

   a. Mediante la consola de AWS RAM, acepte la base de datos que se haya compartido desde la cuenta externa (cuenta B). Para obtener más información, consulte Aceptación de una invitación para compartir un recurso de AWS Resource Access Manager.

   b. Cree un rol de IAM para el rastreador. Agregue la política `lakeformation:GetDataAccess` al rol.

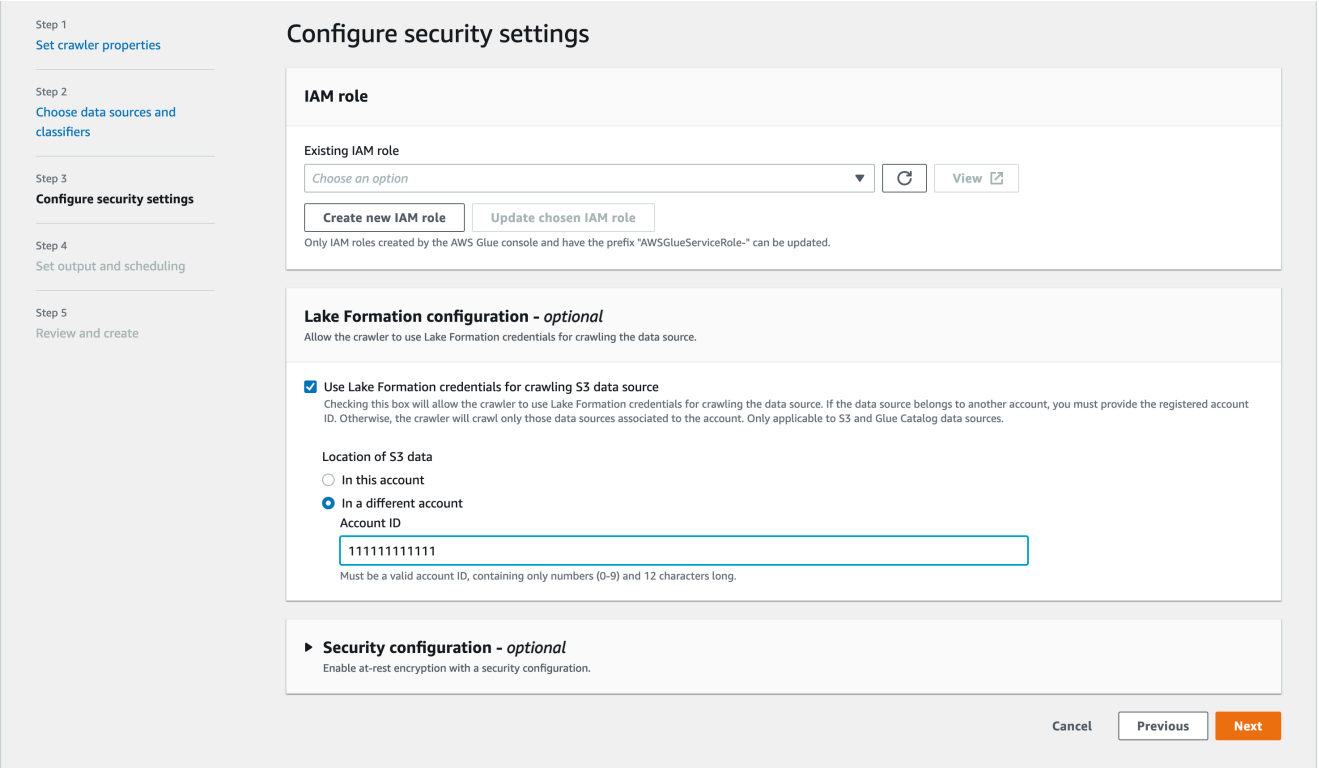
   c. En la consola de Lake Formation (https://console.aws.amazon.com/lakeformation/), conceda permisos de Ubicación de datos sobre la ubicación de Amazon S3 de destino al rol de IAM utilizado para la ejecución del rastreador, de modo que el rastreador pueda leer los datos del destino en Lake Formation. Para obtener más información, consulte Concesión de permisos de ubicación de datos.

   d. Cree un enlace de recurso en la base de datos compartida. Para obtener más información, consulte Creación de un enlace de recurso.

   e. Otorgue al rol del rastreador permisos de acceso (`Create`) en la base de datos compartida y (`Describe`) en el enlace de recurso. El enlace de recurso se especifica en la salida del rastreador.

   f. En la consola de AWS Glue (https://console.aws.amazon.com/glue/), al configurar el rastreador, seleccione la opción Use Lake Formation credentials for crawling Amazon S3 data source (Utilizar credenciales de Lake Formation para rastrear un origen de datos de Amazon S3).

      Para el rastreo entre cuentas, especifique el ID de la Cuenta de AWS donde esté registrada la ubicación de Amazon S3 de destino en Lake Formation. Para el rastreo en cuenta, el campo accountId es opcional.



AWS CLI

```
aws glue --profile demo create-crawler --debug --cli-input-json '{
    "Name": "prod-test-crawler",
    "Role": "arn:aws:iam::111122223333:role/service-role/AWSGlueServiceRole-prod-
test-run-role",
    "DatabaseName": "prod-run-db",
    "Description": "",
    "Targets": {
    "S3Targets":[
                {
                 "Path": "s3://crawl-testbucket"
                }
                ]
                },
   "SchemaChangePolicy": {
      "UpdateBehavior": "LOG",
      "DeleteBehavior": "LOG"
  },
  "RecrawlPolicy": {
    "RecrawlBehavior": "CRAWL_EVERYTHING"
  },
```

```
  "LineageConfiguration": {
    "CrawlerLineageSettings": "DISABLE"
  },
  "LakeFormationConfiguration": {
    "UseLakeFormationCredentials": true,
    "AccountId": "111111111111"
  },
  "Configuration": {
           "Version": 1.0,
           "CrawlerOutput": {
             "Partitions": { "AddOrUpdateBehavior": "InheritFromTable" },
             "Tables": {"AddOrUpdateBehavior": "MergeNewColumns" }
           },
           "Grouping": { "TableGroupingPolicy": "CombineCompatibleSchemas" }
         },
  "CrawlerSecurityConfiguration": "",
  "Tags": {
    "KeyName": ""
  }
}'
```

Note

- Un rastreador que utilice credenciales de Lake Formation solo puede rastrear destinos de Amazon S3 y el Catálogo de datos.
- En el caso de destinos que utilicen el suministro de credenciales de Lake Formation, las ubicaciones de Amazon S3 subyacentes deben pertenecer al mismo bucket. Por ejemplo, los clientes pueden utilizar varios destinos (s3://bucket1/carpeta1, s3://bucket1/carpeta2) siempre que todas las ubicaciones de destino estén en el mismo bucket (bucket1). No se permite especificar buckets diferentes (s3://bucket1/carpeta1, s3://bucket2/carpeta2).
- Actualmente, para los rastreadores de destinos del Catálogo de datos, solo se permite un único destino de catálogo con una sola tabla de catálogo.

Tutorial: agregar un rastreador de AWS Glue

Para este escenario de AWS Glue, se le pedirá que analice los datos de llegada de las principales compañías aéreas para calcular la popularidad de los aeropuertos de salida mes a mes. Tiene datos

de vuelos para el año 2016 en formato CSV almacenado en Amazon S3. Antes de transformar y analizar los datos, catalogue sus metadatos en AWS Glue Data Catalog.

En este tutorial, agregaremos un rastreador que deduce metadatos de estos registros de vuelo en Amazon S3 y crea una tabla en el Data Catalog.

Temas

- Requisitos previos
- Paso 1: agregar un rastreador
- Paso 2: ejecutar el rastreador
- Paso 3: ver objetos del AWS Glue Data Catalog

Requisitos previos

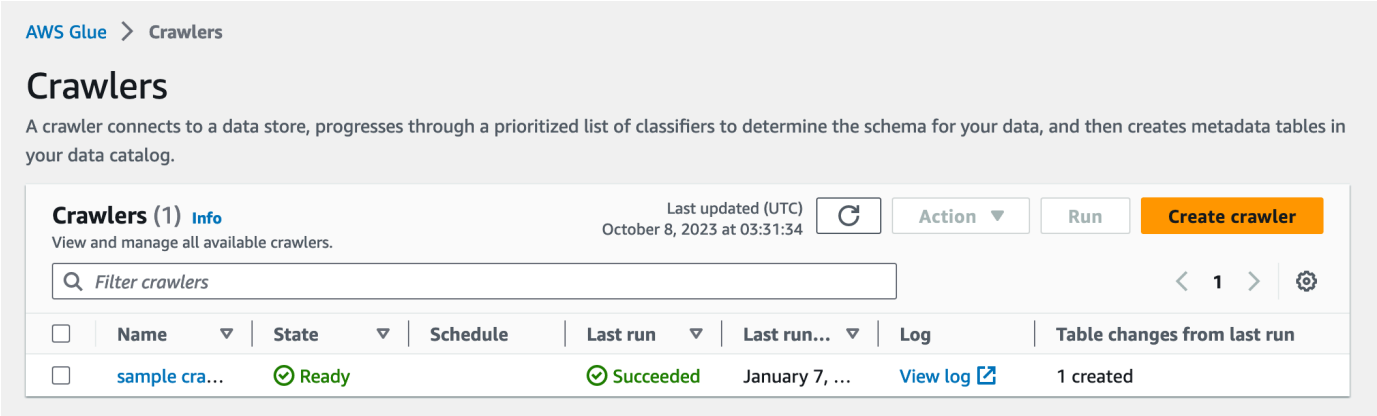
En este tutorial se supone que usted tiene una cuenta de AWS y acceso a AWS Glue.

Paso 1: agregar un rastreador

Siga estos pasos para configurar y ejecutar un rastreador que extraiga los metadatos de un archivo CSV almacenado en Amazon S3.

Para crear un rastreador que lea archivos almacenados en Amazon S3

1. En la consola del servicio AWS Glue, en el menú de la izquierda, elija Crawlers (Rastreadores).
2. En la página Rastreadores, elija Agregar rastreador. Esto inicia una serie de páginas que le solicitan los detalles del rastreador.



3. En el campo Crawler name (Nombre del rastreador), ingrese **`Flights Data Crawler`** y, a continuación, elija Next (Siguiente).

Los rastreadores invocan clasificadores para inferir el esquema de sus datos. En este tutorial se utiliza el clasificador integrado para CSV de forma predeterminada.

4. Para el tipo de origen de rastreador, elija Data stores (Almacenes de datos) y luego elija Next (Siguiente).
5. Ahora apuntaremos el rastreador a sus datos. En la página Add a data store (Agregar un almacén de datos), elija el almacén de datos de Amazon S3. Este tutorial no utiliza una conexión, así que deje el campo Connection (Conexión) en blanco si está visible.

   Para la opción Crawl data in (Rastrear los datos en), elija Specified path in another account (Ruta especificada en otra cuenta). Luego, para Include path (Incluir ruta), ingrese la ruta donde el rastreador puede encontrar los datos de vuelos, que es **`s3://crawler-public-us-east-1/flight/2016/csv`**. Después de introducir la ruta, el título de este campo cambia a Include path (Incluir ruta). Elija Siguiente.
6. Puede rastrear varios almacenes de datos con un único rastreador. Sin embargo, en este tutorial, estamos usando solo un único almacén de datos, así que elija No, y luego Next (Siguiente).
7. El rastreador necesita permisos para acceder al almacén de datos y crear objetos en el AWS Glue Data Catalog. Para configurar estos permisos, elija Create an IAM role (Crear un rol de IAM). El nombre del rol de IAM comienza con `AWSGlueServiceRole-` y, en el campo, ingrese la última parte del nombre del rol. Ingrese **`CrawlerTutorial`** y, a continuación, elija Next (Siguiente).

   > **Note**
   >
   > Para crear un rol de IAM, el usuario de AWS debe tener permisos `CreateRole`, `CreatePolicy` y `AttachRolePolicy`.

   El asistente crea un rol de IAM denominado `AWSGlueServiceRole-CrawlerTutorial`, asocia la política administrada de AWS, `AWSGlueServiceRole`, a este rol y agrega una política en línea que permite el acceso de lectura a la ubicación `s3://crawler-public-us-east-1/flight/2016/csv` de Amazon S3.
8. Cree una programación para el rastreador. Para Frequency (Frecuencia), elija Run on demand (Ejecutar bajo demanda), y luego elija Next (Siguiente).

9. Los rastreadores crean tablas en su Data Catalog. Una base de datos de en una base de datos del Data Catalog. Primero, elija Add database (Agregar base de datos) para crear una base de datos. En la ventana emergente, ingrese **`test-flights-db`** para el nombre de la base de datos y, a continuación, elija Create (Crear).

   Luego, ingrese **`flights`** para Prefix added to tables (Prefijo agregado a las tablas). Utilice los valores predeterminados para el resto de las opciones y elija Next (Siguiente).

10. Compruebe las opciones elegidas en el asistente Add crawler (Agregar rastreador). Si detecta algún error, puede elegir Back (Atrás) para volver a las páginas anteriores y realizar cambios.

    Después de haber revisado la información, elija Finish (Finalizar) para crear el rastreador.

Paso 2: ejecutar el rastreador

Después de crear un rastreador, el asistente lo envía a la página Crawlers view (Visualización de rastreadores). Dado que creó un rastreador con una programación bajo demanda, se le ofrece la opción de ejecutar el rastreador.

Para ejecutar un rastreador

1. El banner situado cerca de la parte superior de esta página le permite saber que se creó el rastreador y le pregunta si desea ejecutarlo ahora. Elija Run it now (Ejecutarlo ahora) para ejecutar el rastreador.

   El banner cambia para mostrar los mensajes “Attempting to run (Intento de ejecución)” and “Running (Ejecución)” para el rastreador. Una vez que el rastreador comienza a ejecutarse, el banner desaparece y la visualización del rastreador se actualiza para mostrar un estado Starting (Inicio) para el rastreador. Después de un minuto, puede hacer clic en el ícono Refresh (Actualizar) para actualizar el estado del rastreador que se muestra en la tabla.

2. Cuando se completa el rastreador, aparece un nuevo banner que describe los cambios realizados por el rastreador. Puede elegir el enlace test-flights-db (probar base de datos de vuelos) para ver los objetos del Data Catalog.

Paso 3: ver objetos del AWS Glue Data Catalog

El rastreador lee los datos en la ubicación de origen y crea tablas en el Data Catalog. Una tabla es una definición de metadatos que representa sus datos e incluye el esquema de datos. Las tablas del

Data Catalog no contienen datos. En su lugar, se utilizan estas tablas como origen o destino en una definición de trabajo.

Para ver los objetos del Data Catalog creados por el rastreador

1. En el panel de navegación de la izquierda, en Data Catalog, elija Databases (Bases de datos). Aquí puede ver la base de datos de `flights-db` que crea el rastreador.
2. En el panel de navegación de la izquierda, Data catalog y luego en Databases (Bases de datos), elija Tables (Tablas). Aquí puede ver la tabla de `flightscsv` que crea el rastreador. Si elige el nombre de la tabla, puede ver la configuración, los parámetros y las propiedades de la tabla. Si se desplaza hacia abajo en esta vista, puede ver el esquema, que es información sobre las columnas y los tipos de datos de la tabla.
3. Si elige View partitions (Ver particiones) en la página de vista de tabla, puede ver las particiones creadas para los datos. La primera columna es la clave de partición.

Cómo definir los metadatos manualmente

El Catálogo de datos de AWS Glue es un repositorio central que almacena los metadatos sobre los orígenes y conjuntos de datos. Si bien un rastreador puede rastrear y completar metadatos automáticamente para los orígenes de datos compatibles, hay determinadas situaciones en las que es posible que deba definir los metadatos manualmente en el Catálogo de datos:

- Formatos de datos no compatibles: Si tiene orígenes de datos que no son compatibles con el rastreador, deberá definir los metadatos correspondientes a esos orígenes de datos en el catálogo de forma manual.
- Requisitos de metadatos personalizados: El Rastreador de AWS Glue infiere los metadatos en función de reglas y convenciones predefinidas. Si tiene requisitos de metadatos específicos que no están cubiertos por los metadatos deducidos mediante el Rastreador de AWS Glue, puede definir los metadatos manualmente para adaptarlos a sus necesidades.
- Gobernanza y estandarización de los datos: En algunos casos, es posible que quiera tener un mayor control sobre las definiciones de los metadatos por motivos de gobernanza, cumplimiento o seguridad de los datos. Al definir los metadatos manualmente, se asegura de que estos cumplan con las normas y políticas de su organización.
- Marcador de posición para la futura ingesta de datos: Si tiene orígenes de datos que no están disponibles o accesibles de forma inmediata, puede crear tablas de esquemas vacías a modo de

marcadores de posición. Una vez que los orígenes de datos estén disponibles, podrá completar las tablas con los datos reales y, al mismo tiempo, mantener la estructura predefinida.

Para definir los metadatos manualmente, puede usar la consola de AWS Glue, la consola de Lake Formation, la API de AWS Glue o la AWS Command Line Interface (AWS CLI). Puede crear bases de datos, tablas y particiones, y especificar las propiedades de los metadatos, como los nombres de las columnas, los tipos de datos, las descripciones y otros atributos.

Creación de bases de datos

Las bases de datos se utilizan para organizar tablas de metadatos en el AWS Glue. Al definir una tabla en el AWS Glue Data Catalog, puede añadirla a una base de datos. Una tabla puede estar en solo una base de datos.

Su base de datos puede contener tablas que definan datos de muchos almacenes de datos diferentes. Estos datos pueden incluir objetos de Amazon Simple Storage Service (Amazon S3) y tablas relacionales de Amazon Relational Database Service.

> **Note**
>
> Al eliminar una base de datos del Catálogo de datos de AWS Glue, también se eliminan todas las tablas de la base de datos.

Para ver la lista de bases de datos, inicie sesión en AWS Management Console y abra la consola de AWS Glue en https://console.aws.amazon.com/glue/. Seleccione Databases (Bases de datos) y, a continuación, seleccione un nombre de base de datos en la lista para ver los detalles.

En la pestaña Databases (Bases de datos) de la consola de AWS Glue, se pueden añadir, editar y eliminar bases de datos:

- Para crear una nueva base de datos, seleccione Add database (Añadir base de datos) e indique un nombre y una descripción. Para ofrecer compatibilidad con otros almacenes de metadatos, como Apache Hive, el nombre se incorpora en minúsculas.

> **Note**
>
> Si tiene previsto obtener acceso a la base de datos desde Amazon Athena, proporcione un nombre únicamente con caracteres alfanuméricos y guiones bajos. Para obtener más información, consulte Nombres de Athena.

- Para editar la descripción de una base de datos, marque la casilla situada junto al nombre de la base de datos y seleccione Edit database (Editar base de datos).
- Para eliminar una base de datos, seleccione la casilla situada junto al nombre de la base de datos y elija Remove (Eliminar).
- Para ver la lista de tablas que contiene la base de datos, elija el nombre de la base de datos; todas las tablas aparecerán enumeradas en las propiedades de la base de datos.

Para cambiar la base de datos de escritura de un rastreador, tendrá que modificar la definición del rastreador. Para obtener más información, consulte Uso de rastreadores para completar el Catálogo de datos .

Enlaces de recursos a base de datos

La consola de AWS Glue se ha actualizado recientemente. La versión actual de la consola no admite enlaces de recursos a base de datos.

El Catálogo de datos también puede contener enlaces de recursos a bases de datos. Un enlace de recursos a bases de datos es un enlace a una base de datos local o compartida. En la actualidad, puede crear enlaces de recursos solo en AWS Lake Formation. Después de crear un enlace de recurso a una base de datos, puede utilizar el nombre del enlace de recursos donde quiera que utilice el nombre de la base de datos. Junto con las bases de datos que posee o que se comparten con usted, los enlaces de recursos a base de datos son devueltos por `glue:GetDatabases()` y aparecerán como entradas en la página Databases (Bases de datos) de la consola de AWS Glue.

El Catálogo de datos también puede contener enlaces de recursos a tablas.

Para obtener más información acerca de los enlaces de recursos, consulte Creación de enlaces de recursos en la Guía para desarrolladores de AWS Lake Formation.

Creación de tablas

Aunque ejecutar un rastreador es el método recomendado para hacer un inventario de los datos incluidos en sus almacenes de datos, puede añadir tablas de metadatos al AWS Glue Data Catalog manualmente. Este enfoque le permite tener más control sobre las definiciones de los metadatos y personalizarlas de acuerdo con sus requisitos específicos.

También puede agregar tablas al Catálogo de datos manualmente de las siguientes formas:

- Utilice la consola de AWS Glue para crear de forma manual una tabla en el AWS Glue Data Catalog. Para obtener más información, consulte Trabajo con tablas en la consola de AWS Glue.
- Use la operación `CreateTable` en AWS Glue API para crear una tabla en AWS Glue Data Catalog. Para obtener más información, consulte CreateTable acción (Python: create_table).
- Utilice plantillas de AWS CloudFormation. Para obtener más información, consulte AWS CloudFormation para AWS Glue.

Al definir una tabla de forma manual mediante la consola o una API, especifica el esquema de tabla y el valor de un campo de clasificación que indica el tipo y el formato de los datos de la fuente de datos. Si un rastreador crea la tabla, el formato de los datos y el esquema se determinan por un clasificador personalizado o un clasificador integrado. Para obtener información adicional acerca de cómo crear una tabla mediante la consola de AWS Glue, consulte Trabajo con tablas en la consola de AWS Glue.

Temas

- Particiones de la tabla
- Enlaces de recursos a tabla
- Actualización de tablas del Catálogo de datos creadas de forma manual mediante rastreadores
- Propiedades de la tabla del catálogo de datos
- Trabajo con tablas en la consola de AWS Glue
- Trabajar con índices de partición en AWS Glue

Particiones de la tabla

Una definición de tabla de AWS Glue de una carpeta de Amazon Simple Storage Service (Amazon S3) puede describir una tabla con particiones. Por ejemplo, para mejorar el desempeño de las consultas, una tabla con particiones podría separar datos mensuales en diferentes archivos con el

nombre del mes como clave. En AWS Glue, las definiciones de tabla incluyen la clave de partición de una tabla. Cuando AWS Glue evalúa los datos de carpetas de Amazon S3 para catalogar una tabla, determina si se agrega una tabla individual o con particiones.

Puede crear índices de partición en una tabla para obtener un subconjunto de las particiones en lugar de cargar todas las particiones de la tabla. Para obtener más información sobre cómo trabajar con índices de partición, consulte Trabajar con índices de partición en AWS Glue.

Todas las condiciones siguientes deben cumplirse para que AWS Glue cree una tabla con particiones para una carpeta de Amazon S3:

- Los esquemas de los archivos son similares, según determine AWS Glue.
- El formato de datos de los archivos es el mismo.
- El formato compresión de los archivos es el mismo.

Por ejemplo, es posible que posea un bucket de Amazon S3 denominado `my-app-bucket`, en el que almacene datos de aplicaciones de ventas de iOS y Android. Los datos se distribuyen en particiones por año, mes y día. Los archivos de datos para ventas iOS y Android tienen el mismo esquema, formato de datos y formato de compresión. En AWS Glue Data Catalog el rastreador de AWS Glue crea una definición de tabla con claves de partición para año, mes y día.

En la siguiente lista de Amazon S3 para `my-app-bucket` se muestran algunas de las particiones. El símbolo = se utiliza para asignar valores de clave de partición.

```
my-app-bucket/Sales/year=2010/month=feb/day=1/iOS.csv
my-app-bucket/Sales/year=2010/month=feb/day=1/Android.csv
my-app-bucket/Sales/year=2010/month=feb/day=2/iOS.csv
my-app-bucket/Sales/year=2010/month=feb/day=2/Android.csv
...
my-app-bucket/Sales/year=2017/month=feb/day=4/iOS.csv
my-app-bucket/Sales/year=2017/month=feb/day=4/Android.csv
```

Enlaces de recursos a tabla

La consola de AWS Glue se ha actualizado recientemente. La versión actual de la consola no admite enlaces de recursos a tabla.

El Catálogo de datos también puede contener enlaces de recursos a tablas. Un enlace de recursos a tablas es un enlace a una tabla local o compartida. En la actualidad, puede crear enlaces de recursos solo en AWS Lake Formation. Después de crear un enlace de recurso a una tabla, puede utilizar el nombre del enlace de recursos donde quiera que utilice el nombre de la tabla. Junto con las tablas que posee o que se comparten con usted, los enlaces de recursos a tablas son devueltos por `glue:GetTables()` y aparecerán como entradas en la página Tables (Tablas) de la consola de AWS Glue.

El Catálogo de datos también puede contener enlaces de recursos a tablas.

Para obtener más información acerca de los enlaces de recursos, consulte Creación de enlaces de recursos en la Guía para desarrolladores de AWS Lake Formation.

Actualización de tablas del Catálogo de datos creadas de forma manual mediante rastreadores

Es posible que desee crear tablas de AWS Glue Data Catalog de forma manual y, a continuación, mantenerlas actualizadas con rastreadores de AWS Glue. Los rastreadores que se ejecutan en una programación pueden añadir nuevas particiones y actualizar las tablas con cualquier cambio de esquema. Esto también se aplica a tablas migradas desde un metaalmacén de Apache Hive.

Para ello, cuando defina un rastreador, en lugar de especificar un almacén de datos o más como origen de un rastreador, especifique una o más tablas existentes del Catálogo de datos. El rastreador rastrea los almacenes de datos especificados por las tablas del catálogo. En este caso, no se crean tablas nuevas. En su lugar, las tablas que usted crea de forma manual se actualizan.

A continuación se muestran otros motivos por los que podría desear crear tablas de catálogos de forma manual y especificar tablas de catálogos como la fuente del rastreador:

- Desea elegir el nombre de la tabla de catálogo y no confía en el algoritmo de denominación de tablas de catálogos.
- Desea evitar que se creen tablas nuevas en caso de que los archivos que tengan un formato que podría interrumpir la detección de particiones se guarden de forma errónea en la ruta de la fuente de datos.

Para obtener más información, consulte Paso 2: elegir orígenes de datos y clasificadores.

Propiedades de la tabla del catálogo de datos

Las propiedades o los parámetros de la tabla, tal como se les conoce en la AWS CLI, son cadenas de valores y claves no validadas. Puede establecer sus propias propiedades en la tabla para permitir

usos del catálogo de datos fuera de AWS Glue. Otros servicios que utilizan el Catálogo de datos también pueden hacerlo. AWS Glue establece algunas propiedades de la tabla al ejecutar trabajos o rastreadores. A menos que se describa lo contrario, estas propiedades son para uso interno, no se admite que sigan existiendo en su forma actual ni el comportamiento del producto si estas propiedades se cambian manualmente.

Para obtener más información sobre las propiedades de tabla establecidas por los rastreadores de AWS Glue, consulte the section called “Parámetros establecidos en las tablas del catálogo de datos por el rastreador”.

Trabajo con tablas en la consola de AWS Glue

Una tabla en el AWS Glue Data Catalog es la definición de metadatos que representa los datos en un almacén de datos. Puede crear tablas al ejecutar un rastreador, o bien puede crear una tabla manualmente en la consola de AWS Glue. En la lista Tablas en la consola de AWS Glue se muestran los valores de los metadatos de su tabla. Puede usar definiciones de tabla para especificar orígenes y destinos al crear trabajos de ETL (extracción, transformación y carga).

> **Note**
>
> Con los cambios recientes en la consola de administración de AWS, es posible que tenga que modificar sus roles de IAM existentes para obtener permiso de `SearchTables`. Para la creación de nuevos roles, se agregó de forma predeterminada el permiso de la API de `SearchTables`.

Para comenzar, inicie sesión en AWS Management Console y abra la consola de AWS Glue en https://console.aws.amazon.com/glue/. Elija la pestaña Tablas y use el botón Agregar tablas para crear tablas con un rastreador o escribiendo atributos manualmente.

Adición de tablas en la consola

A fin de usar un rastreador para agregar tablas, elija Agregar tablas y Agregar tablas mediante un rastreador. A continuación, siga las instrucciones en el asistente Adición de rastreadores. Cuando se ejecuta el rastreador, se agregan las tablas al AWS Glue Data Catalog. Para obtener más información, consulte Uso de rastreadores para completar el Catálogo de datos .

Si conoce los atributos necesarios para crear una definición de tabla de Amazon Simple Storage Service (Amazon S3) en su Data Catalog puede crearla con el asistente de tabla. Elija Agregar tablas, Agregar tabla manualmente y siga las instrucciones en el asistente Agregar tabla.

Al agregar una tabla manualmente a través de la consola, tenga en cuenta lo siguiente:

- Si tiene previsto obtener acceso a la tabla desde Amazon Athena, proporcione un nombre únicamente con caracteres alfanuméricos y guiones bajos. Para obtener más información, consulte Nombres de Athena.
- La ubicación de sus datos de origen debe ser una ruta de Amazon S3.
- El formato de datos de los datos debe coincidir con uno de los formatos que aparecen en el asistente. La clasificación correspondiente, SerDe, y otras propiedades de tabla se rellenan automáticamente según el formato elegido. Puede definir tablas con los siguientes formatos:

  Avro

  Formato binario JSON Apache Avro.

  CSV

  Valores separados por caracteres. También puede especificar el delimitador de coma, barra vertical, punto y coma, tabulador o Ctrl-A.

  JSON

  JavaScript Object Notation, notación de objetos de JavaScript.

  XML

  Formato de lenguaje de marcado extensible. Especifique la etiqueta XML que define una fila en los datos. Las columnas se definen dentro de etiquetas de fila.

  Parquet

  Almacenamiento en columnas de Apache Parquet.

  ORC

  Formato archivo Optimized Row Columnar (ORC). Formato diseñado para almacenar de forma eficiente los datos de Hive.
- Puede definir una clave de partición para la tabla.
- Actualmente, las tablas con particiones que crea con la consola no se pueden usar en los trabajos de ETL.

Atributos de tabla

A continuación se muestran algunos atributos importantes de su tabla:

Nombre

El nombre se determina al crearse la tabla y no puede cambiarlo. Puede hacer referencia a un nombre de tabla en muchas operaciones de AWS Glue.

Base de datos

El objeto contenedor donde reside su tabla. Este objeto contiene una organización de sus tablas que existe en el AWS Glue Data Catalog y puede diferir de una organización en su almacén de datos. Al eliminar una tabla, todas las tablas incluidas en la base de datos también se eliminan del Data Catalog.

Descripción

Descripción de la tabla. Puede escribir una descripción para ayudarle a entender el contenido de la tabla.

Formato de tabla

Especifique la creación de una tabla estándar de AWS Glue o de una tabla en formato Apache Iceberg.

Habilitar la compactación

Seleccione Habilitar la compactación para compactar objetos pequeños de Amazon S3 de la tabla y convertirlos en objetos más grandes.

Rol de IAM

Para ejecutar la compactación, el servicio asume un rol de IAM en su nombre. Puede elegir un rol de IAM mediante el menú desplegable. Asegúrese de que el rol tenga los permisos necesarios para habilitar la compactación.

Para obtener más información sobre los permisos necesarios para este rol de IAM, consulte Requisitos previos para la optimización de tablas .

Ubicación

El señalizador a la ubicación de los datos en un almacén de datos que representa esta definición de tabla.

Clasificación

Un valor de categorización proporcionado cuando se creó la tabla. Normalmente, este se escribe al ejecutarse un rastreador y especifica el formato de los datos de origen.

Última actualización

La hora y la fecha (UTC) en que se actualizó esta tabla en el Data Catalog.

Fecha agregada

La hora y la fecha (UTC) en que se agregó esta tabla al Data Catalog.

Obsoleto

Si AWS Glue descubre que una tabla en el Data Catalog ya no existe en su almacén de datos original, marca la tabla como obsoleta en el catálogo de datos. Si ejecuta un flujo de trabajo que hace referencia a una tabla obsoleta, podría producirse un error en el flujo de trabajo. Edite trabajos que hagan referencia a tablas obsoletas para quitarlas como orígenes y destinos. Recomendamos que elimine las tablas obsoletas cuando ya no sean necesarias.

Connection

Si AWS Glue requiere una conexión a su almacén de datos, el nombre de la conexión se asocia a la tabla.

Visualización y edición de los detalles de la tabla

Para ver los detalles de una tabla existente, elija el nombre de tabla de la lista y, a continuación, elija Acción, Ver detalles.

Entre los detalles de la tabla se incluyen propiedades de su tabla y su esquema. Esta vista muestra el esquema de la tabla, incluidos los nombres de columna en el orden definido para la tabla, los tipos de datos y las columnas con clave para las particiones. Si una columna es un tipo complejo, puede elegir Ver propiedades para mostrar detalles de la estructura de ese campo, como se muestra en el siguiente ejemplo:

```
{
"StorageDescriptor":
    {
      "cols": {
         "FieldSchema": [
           {
             "name": "primary-1",
             "type": "CHAR",
             "comment": ""
           },
```

```
            {
              "name": "second ",
              "type": "STRING",
              "comment": ""
            }
          ]
      },
      "location": "s3://aws-logs-111122223333-us-east-1",
      "inputFormat": "",
      "outputFormat": "org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat",
      "compressed": "false",
      "numBuckets": "0",
      "SerDeInfo": {
           "name": "",
           "serializationLib": "org.apache.hadoop.hive.serde2.OpenCSVSerde",
           "parameters": {
               "separatorChar": "|"
            }
      },
      "bucketCols": [],
      "sortCols": [],
      "parameters": {},
      "SkewedInfo": {},
      "storedAsSubDirectories": "false"
    },
    "parameters": {
       "classification": "csv"
    }
}
```

Para obtener más información acerca de las propiedades de una tabla, como `StorageDescriptor`, consulte StorageDescriptor estructura.

Para cambiar el esquema de una tabla, elija Editar esquema para agregar y quitar columnas, cambiar nombres de columna y cambiar tipos de datos.

Para comparar diferentes versiones de una tabla, incluido su esquema, elija Comparar versiones para ver una comparación paralela de dos versiones del esquema para una tabla. Para obtener más información, consulte Comparación de las versiones del esquema de la tabla .

Para mostrar los archivos que componen una partición de Amazon S3, elija Ver partición. Para las tablas de Amazon S3, la columna Clave muestra las claves de partición que se usan para particionar la tabla en el almacén de datos de origen. La creación de particiones es una forma de dividir una

tabla en partes relacionadas según los valores de una columna de clave, tales como fecha, ubicación o departamento. Para obtener más información acerca de las particiones, busque en Internet información acerca de la "creación de particiones Hive".

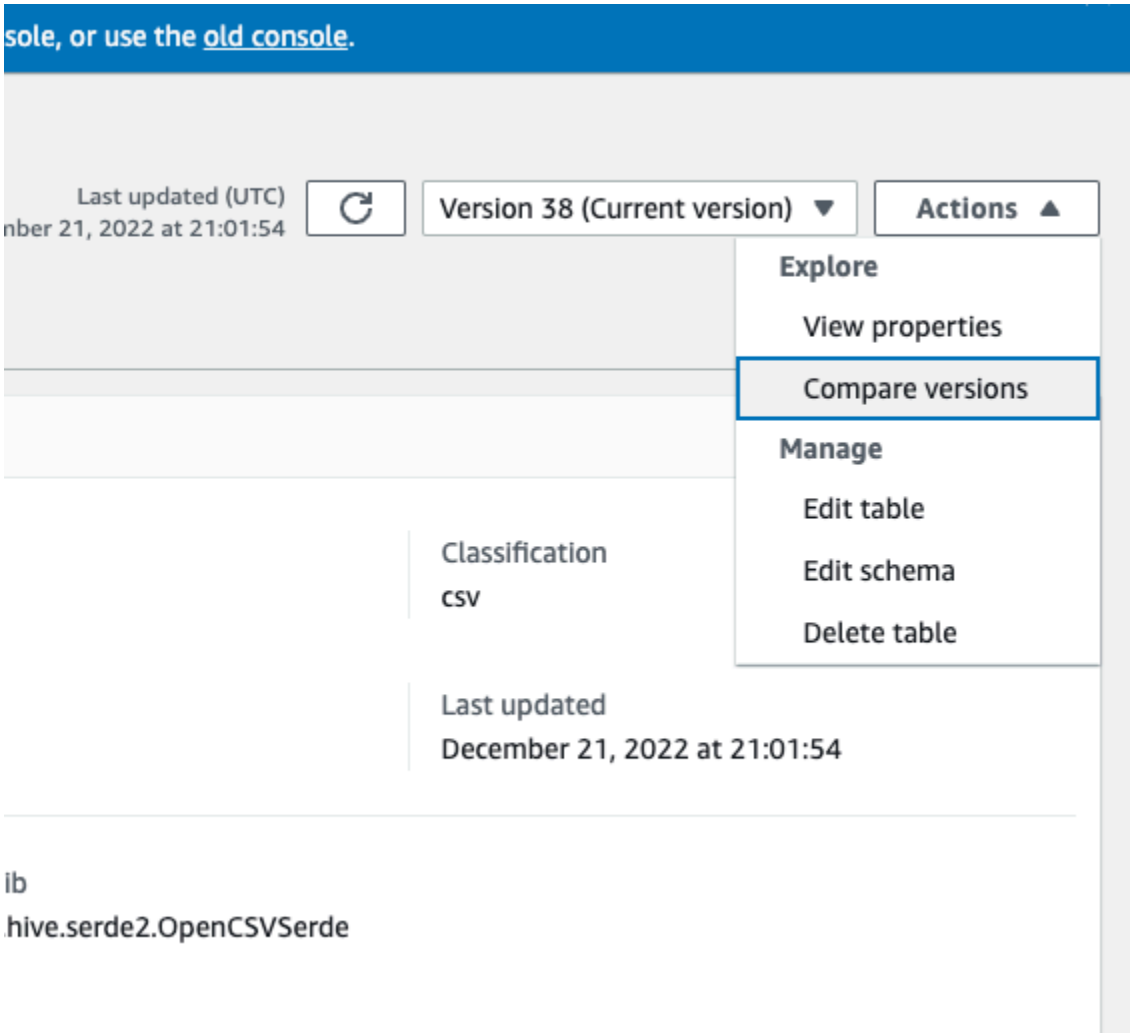
> **Note**
>
> Para obtener instrucciones paso a paso para ver los detalles de una tabla, consulte el tutorial Explorar tabla en la consola.

Comparación de las versiones del esquema de la tabla

Cuando compara dos versiones de esquemas de tablas, puede comparar los cambios en las filas anidadas al expandir y contraer las filas anidadas, puede comparar los esquemas de dos versiones y ver las propiedades de las tablas lado a lado.

Para comparar las versiones

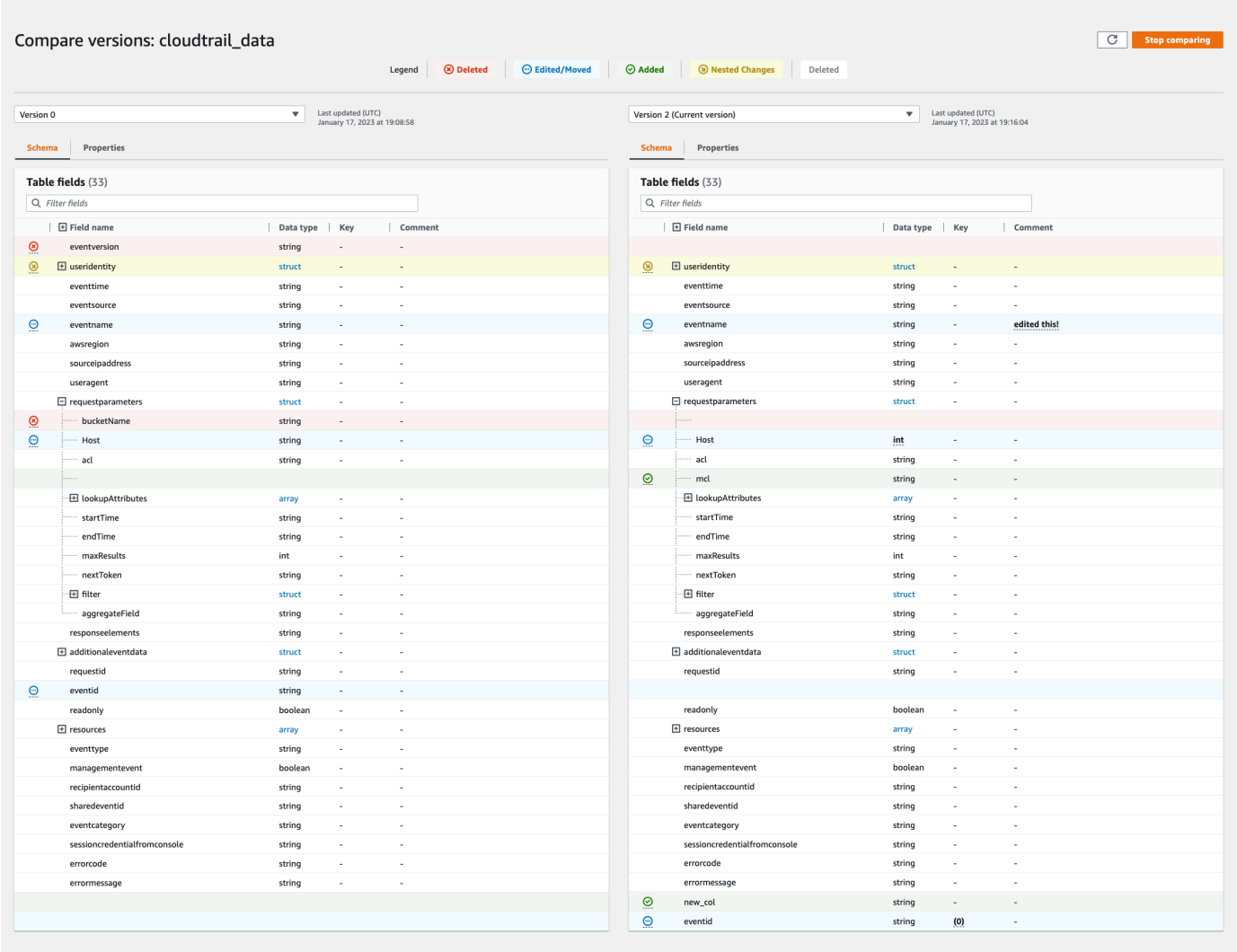
1. En la consola de AWS Glue, seleccione Tablas, Acciones y, a continuación, elija Comparar versiones.



2. Elija una versión para comparar mediante el menú desplegable de versiones. Cuando compare esquemas, la pestaña Esquema aparece resaltada en naranja.
3. Cuando compare tablas entre dos versiones, los esquemas de las tablas se muestran en la parte izquierda y derecha de la pantalla. Esto le permite visualizar los cambios al comparar los campos de Nombre de columna, tipo de datos, clave y comentario uno al lado del otro. Cuando se produce un cambio, aparece un icono de color que muestra el tipo de cambio realizado.

   - Eliminada: se muestra un icono rojo que indica dónde se quitó la columna de una versión anterior del esquema de la tabla.
   - Editada o movida: se muestra un icono azul que indica dónde se modificó o movió la columna en una versión más reciente del esquema de la tabla.
   - Agregado: se muestra un icono verde que indica dónde se agregó una columna a una versión más reciente del esquema de la tabla.

- Cambios anidados: se muestra un icono amarillo que indica dónde contiene cambios la columna anidada. Elija la columna para expandirla y ver las columnas que se eliminaron, editaron, movieron o agregaron.



4. Utilice la barra de búsqueda de campos filtrados para mostrar los campos en función de los caracteres que introduzca aquí. Si introduce un nombre de columna en cualquier versión de la tabla, los campos filtrados se muestran en ambas versiones de la tabla para mostrarle dónde se produjeron los cambios.
5. Para comparar propiedades, elija la pestaña de Propiedades.
6. Para detener la comparación de versiones, elija Detener comparación para volver a la lista de tablas.

Trabajar con índices de partición en AWS Glue

Con el tiempo, cientos de miles de particiones se agregan a una tabla. La API GetPartitions se utiliza para buscar las particiones en la tabla. La API devuelve particiones que coinciden con la expresión proporcionada en la solicitud.

Utilicemos, a modo de ejemplo, una tabla sales_data (datos de servicio), la cual está dividida en las secciones Country (País), Category (Categoría), Year (Año), Month (Mes) y creationDate (Fecha de creación). Para obtener los datos de ventas de todos los objetos que se vendieron en la categoría Books (Libros) durante el 2020, después del 2020-08-15, tiene que crear una solicitud de `GetPartitions` con la expresión “Category = ‘Books’ and creationDate > ‘2020-08-15’” en el catálogo de datos.

Si no hay índices de partición presentes en la tabla, AWS Glue carga todas las particiones de la tabla y, a continuación, filtra las particiones cargadas con la expresión de consulta proporcionada por el usuario en la solicitud `GetPartitions`. La consulta tarda más tiempo en ejecutarse a medida que aumenta el número de particiones en una tabla sin índices. Con un índice, la consulta `GetPartitions` intentará obtener un subconjunto de las particiones en lugar de cargar todas las particiones en la tabla.

Temas

- Acerca de los índices de partición
- Creación de una tabla con índices de partición
- Agregado de un índice de partición a una tabla existente
- Descripción de índices de partición en una tabla
- Limitaciones al uso de índices de partición
- Uso de índices para una llamada GetPartitions optimizada
- Integración con motores

Acerca de los índices de partición

Cuando crea un índice de partición, especifica una lista de claves de partición que ya existen en una tabla determinada. El índice de partición es una sublista de claves de partición definidas en la tabla. Se puede crear un índice de partición en cualquier permutación de claves de partición definidas en la tabla. Para la tabla sales_data anterior, los índices posibles son (país, categoría, fecha de creación), (país, categoría, año), (país, categoría), (país), (categoría, país, año, mes), etc.

El Data Catalog concatenará los valores de partición en el orden proporcionado en el momento de la creación del índice. El índice se genera de forma continua a medida que se agregan particiones a la tabla. Los índices pueden crearse para los tipos de columna Cadena (string, char y varchar), Numérico (int, biging, long, tinyint y smallint) y Fecha (aaaa-mm-dd).

Tipos de datos compatibles

- Fecha: una fecha en el formato ISO, como `YYYY-MM-DD`. Por ejemplo, la fecha `2020-08-15`. El formato utiliza guiones (-) para separar el año, el mes y el día. El rango de fechas admitido para los lapsos de indexación es de `0000-01-01` a `9999-12-31`.
- Cadena: un literal de cadena entre comillas simples o dobles.
- Char: datos de caracteres de longitud fija, con una longitud especificada comprendida entre 1 y 255 como, por ejemplo, char(10).
- Varchar: datos de caracteres de longitud variable, con una longitud especificada comprendida entre 1 y 65 535 como, por ejemplo, varchar(10).
- Numérico: int, bigint, long, tinyint y smallint

Los índices para los tipos de datos Numérico, Cadena y Fecha admiten el uso de los símbolos **=**, **>**, **>=**, **<** y **<=** entre los operadores. La solución de indexación actualmente solo admite el operador lógico `AND`. Las subexpresiones con los operadores “LIKE (COMO)”, “IN (EN)”, “OR (O)” y “NOT (NO)” se ignoran en la expresión para el filtrado mediante índice. El filtrado de la subexpresión ignorada se realiza en las particiones obtenidas después de aplicar el filtrado de índices.

Para cada partición agregada a una tabla, se crea elemento de índice correspondiente. Para una tabla con particiones “n”, un índice de partición dará como resultado elementos de índice de partición “n”. El índice de partición “m” en la misma tabla dará como resultado elementos de índice de partición “m*n”. Cada elemento del índice de partición se cargará de acuerdo con la política de precios actual de AWS Glue para el almacenamiento del catálogo de datos. Para obtener detalles sobre el precio de los objetos de almacenamiento, consulte Precios de AWS Glue.

Creación de una tabla con índices de partición

Puede crear un índice de partición durante la creación de la tabla. La solicitud `CreateTable` toma una lista de objetos de `PartitionIndex` como entrada. Se puede crear un máximo de 3 índices de partición en una tabla determinada. Cada índice de partición requiere un nombre y una lista de `partitionKeys` definida para la tabla. Los índices creados en una tabla se pueden recuperar con la API `GetPartitionIndexes`

Agregado de un índice de partición a una tabla existente

Para agregar un índice de partición a una tabla existente, se usa la operación `CreatePartitionIndex`. Puede crear un `PartitionIndex` por cada operación `CreatePartitionIndex`. Agregar un índice no afecta la disponibilidad de una tabla, ya que la tabla sigue estando disponible mientras se crean los índices.

El estado del índice para una partición agregada se establece en CREATING (CREACIÓN) y se inicia la creación de los datos del índice. Si el proceso de creación de los índices se realiza correctamente, el estado del índice se actualiza a ACTIVE (ACTIVO) y, en caso de que el proceso no se realice correctamente, el estado del índice se actualiza a FAILED (ERROR). La creación del índice puede presentar errores por varias razones, y puede usar la función `GetPartitionIndexes` para recuperar los detalles del error. Los posibles errores son:

- ENCRYPTED_PARTITITION_ERROR (ERROR DE PARTICICÓN CIFRADA): no se admite la creación de índices en una tabla con particiones cifradas.
- INVALID_PARTITITION_TYPE_DATA_ERROR (ERROR DE DATOS DE TIPO DE PARTICIÓN INVÁLIDOS): se observa cuando el valor `partitionKey` no es un valor válido para el tipo de datos `partitionKey` correspondiente. Por ejemplo: una `partitionKey` con el tipo de datos “int” tiene un valor “foo”.
- MISSING_PARTITITION_VALUE_ERROR (ERROR DE VALOR DE PARTICIÓN FALTANTE): se observa cuando el `partitionValue` para una `indexedKey` no está presente. Esto puede suceder cuando una tabla no se particiona de manera uniforme.
- UNSUPPORTED_PARTITITION_CARACTER_ERROR (ERROR DE CARÁCTER DE PARTICIÓN NO SOPORTADO): se observa cuando el valor de una clave de partición indexada contiene los caracteres \u0000, \u0001 o \u0002
- INTERNAL_ERROR (ERROR INTERNO): se produjo un error interno mientras se creaban los índices.

Descripción de índices de partición en una tabla

Para obtener los índices de partición creados en una tabla, utilice la operación `GetPartitionIndexes`. La respuesta enumera todos los índices de la tabla, junto con el estado actual de cada uno de ellos (el `IndexStatus`).

El `IndexStatus` para un índice de partición será uno de los siguientes:

- `CREATING`: el índice está en proceso de creación y aún no está disponible para usarlo.

- `ACTIVE`: ya se puede utilizar el índice. Las solicitudes pueden utilizar el índice para realizar una consulta optimizada.
- `DELETING`: el índice se está eliminando y ya no se puede utilizar. Un índice en el estado activo se puede eliminar mediante la solicitud `DeletePartitionIndex`, que mueve el estado de ACTIVE (ACTIVO) a DELETING (ELIMINACIÓN).
- `FAILED`: error en la creación del índice en una tabla existente. Cada tabla almacena los últimos 10 índices que presentaron errores.

Las transiciones de estado posibles para los índices creados en una tabla existente son:

- CREACIÓN → ACTIVO → ELIMINACIÓN
- CREACIÓN → ERROR

Limitaciones al uso de índices de partición

Una vez que haya creado un índice de partición, tenga en cuenta estos cambios en la funcionalidad de la tabla y la partición:

Creación de una nueva partición (después del agregado de índices)

Después de crear un índice de partición en una tabla, se validarán las comprobaciones de tipos de datos de claves indexadas de todas las particiones nuevas agregadas a la tabla. Se validará el formato de tipo de datos del valor de partición de las claves indexadas. Si se produce un error en la comprobación del tipo de datos, la operación de creación de partición fallará. Para la tabla sales_data (datos de venta), si se crea un índice para claves (categoría, año) donde la categoría es de tipo `string` y año del tipo `int`, la creación de la nueva partición con un valor de YEAR (AÑO) de “foo” presentará errores.

Una vez habilitados los índices, el agregado de particiones con valores clave indexados que tengan los caracteres U+0000, U+00001 y U+0002 comenzará a fallar.

Actualizaciones de tablas

Una vez creado un índice de partición en una tabla, no puede modificar los nombres de las claves de partición para las claves de partición existentes y no puede cambiar el tipo o el orden de las claves que están registradas con el índice.

Uso de índices para una llamada GetPartitions optimizada

Cuando llame `GetPartitions` en una tabla con un índice, puede incluir una expresión y, si corresponde, el Data Catalog utilizará un índice si es posible. La primera clave del índice debe ser transferida a la expresión para los índices que se van a utilizar en el filtrado. La optimización de índices en el filtrado se aplica en la medida de lo posible. El Data Catalog intenta utilizar la optimización del índice tanto como sea posible, pero en caso de que falte un índice o en caso de un operador no soportado, vuelve a la implementación existente de cargar todas las particiones.

Para la tabla sales_data (datos de venta) anterior, agregaremos el índice [Country (País), Category (Categoría), Year (Año)]. Si “País” no se transfiere en la expresión, el índice registrado no podrá filtrar particiones utilizando índices. Puede agregar hasta 3 índices para soportar varios patrones de consulta.

Tomemos algunas expresiones de ejemplo y veamos cómo funcionan los índices en ellas:

Expressions	Cómo se utilizará el índice
Country = 'US' (País = “EE.UU.”)	El índice se utilizará para filtrar particiones.
Country = 'US' (País = “EE.UU.”) y Category = 'Shoes' (Categoría = ”Zapatos”)	El índice se utilizará para filtrar particiones.
Category = 'Shoes' (Categoría = “Zapatos”)	Los índices no se utilizarán ya que no se proporciona “país” en la expresión. Todas las particiones se cargarán para devolver una respuesta.
Country = 'US' (País = “EE.UU.”) y Category = 'Shoes' (Categoría = ”Zapatos”) y Year > '2018' (Año > “2018”)	El índice se utilizará para filtrar particiones.
Country = 'US' (País = “EE.UU.”) y Category = 'Shoes' (Categoría = ”Zapatos”) y Year > '2018' (Año > “2018”) y month = 2 (mes = 2)	El índice se utilizará para buscar todas las particiones con país = “EE.UU.” y categoría = “zapatos” y año > 2018. A continuación, se realizará el filtrado en la expresión del mes.
País = “EE.UU.” Y Categoría = ”Zapatos” O Año > “2018”	Los índices no se usarán ya que se incluye un operador `OR` en la expresión.

Expressions	Cómo se utilizará el índice
País = “EE.UU.” Y Categoría = “Zapatos” Y (Año = 2017 O Año = “2018”)	El índice se utilizará para buscar todas las particiones con país = “EE.UU.” y categoría = “zapatos”, y luego se realizará el filtrado en la expresión del año.
País en (“EE.UU.”, “UK”) Y Categoría = “Zapatos”	Los índices no se usarán para filtrar ya que el operador `IN` no se soporta actualmente.
País = “EE.UU.”, Y Categoría en (“Zapatos”, “Libros”)	El índice se utilizará para buscar todas las particiones con país = “EE.UU.” y luego se realizará el filtrado en la expresión de la categoría.
País = ‘EE. UU.’ Y Categoría en (‘Zapatos’, ’Libros’) Y (Fecha de creación > ‘2023-9-01’	El índice se utilizará para buscar todas las particiones con País = ‘EE. UU.’ y Fecha de creación > ‘2023-9-01’ y luego se realizará el filtrado en la expresión de la Categoría.

Integración con motores

Redshift Spectrum, Amazon EMR y AWS Glue ETL Spark DataFrames pueden utilizar índices para obtener particiones después de que los índices estén en estado ACTIVE en AWS Glue. Athena y los AWS Glue marcos dinámicos de ETL requieren que siga pasos adicionales si desea utilizar índices para mejorar las consultas.

Habilitación del filtrado de particiones

Para habilitar el filtrado de particiones, se necesita actualizar las propiedades de la tabla tal como se muestra a continuación:

1. En la consola de AWS Glue, en la sección Catálogo de datos, seleccione Tablas.
2. Elija una tabla.
3. En Acciones, seleccione Editar tabla.
4. En Propiedades de la tabla, agregue lo siguiente:
   - Clave: `partition_filtering.enabled`

- Valor: `true`

5. Seleccione Aplicar.

Como alternativa, puede establecer este parámetro al ejecutar la consulta ALTER TABLE SET PROPERTIES (Alterar las propiedades establecidas de la tabla) en Athena.

```
ALTER TABLE partition_index.table_with_index
SET TBLPROPERTIES ('partition_filtering.enabled' = 'true')
```

Integración con otros servicios de AWS

Si bien puede usar los Rastreador de AWS Glue para completar el AWS Glue Data Catalog, hay varios servicios de AWS que pueden integrarse con el catálogo y completarlo automáticamente. En las siguientes secciones se proporciona más información sobre los casos de uso específicos compatibles con los servicios de AWS que pueden completar el Catálogo de datos.

Temas

- AWS Lake Formation
- Amazon Athena

AWS Lake Formation

AWS Lake Formation es un servicio que facilita la configuración de un lago de datos seguro en AWS. Lake Formation se basa en AWS Glue, y ambos comparten el mismo AWS Glue Data Catalog. Puede registrar la ubicación de sus datos de Amazon S3 con Lake Formation y usar la consola de Lake Formation para crear bases de datos y tablas en el Catálogo de datos de AWS Glue, definir políticas de acceso a los datos y auditar el acceso a estos en todo su lago de datos desde un solo lugar. Puede utilizar el control de acceso detallado de Lake Formation para gestionar los recursos del Catálogo de datos existentes y las ubicaciones de datos de Amazon S3.

Si registra los datos con Lake Formation, puede compartir de forma segura los recursos del Catálogo de datos entre las entidades principales de IAM, cuentas de AWS, organizaciones de AWS y unidades organizativas.

Para obtener más información sobre cómo crear recursos en el Catálogo de datos con Lake Formation, consulte Creación de tablas y bases de datos del Catálogo de datos en la Guía para desarrolladores de AWS Lake Formation.

Amazon Athena

Amazon Athena usa el Catálogo de datos para almacenar y recuperar metadatos de tablas para los datos de Amazon S3 en su cuenta de AWS. Los metadatos de la tabla permiten al motor de consultas de Athena saber cómo buscar, leer y procesar los datos que desea consultar.

Para completar el AWS Glue Data Catalog, puede usar directamente las instrucciones de Athena para `CREATE TABLE`. Puede definir y completar manualmente los metadatos del esquema y la partición en el Catálogo de datos sin necesidad de ejecutar un rastreador.

1. En la consola de Athena, cree una base de datos que almacene los metadatos de la tabla en el Catálogo de datos.
2. Use la instrucción `CREATE EXTERNAL TABLE` para definir el esquema del origen de datos.
3. Use la cláusula `PARTITIONED BY` para definir las claves de partición si sus datos están particionados.
4. Use la cláusula `LOCATION` para especificar la ruta de Amazon S3 en la que se almacenan sus archivos de datos reales.
5. Ejecute la instrucción `CREATE TABLE`.

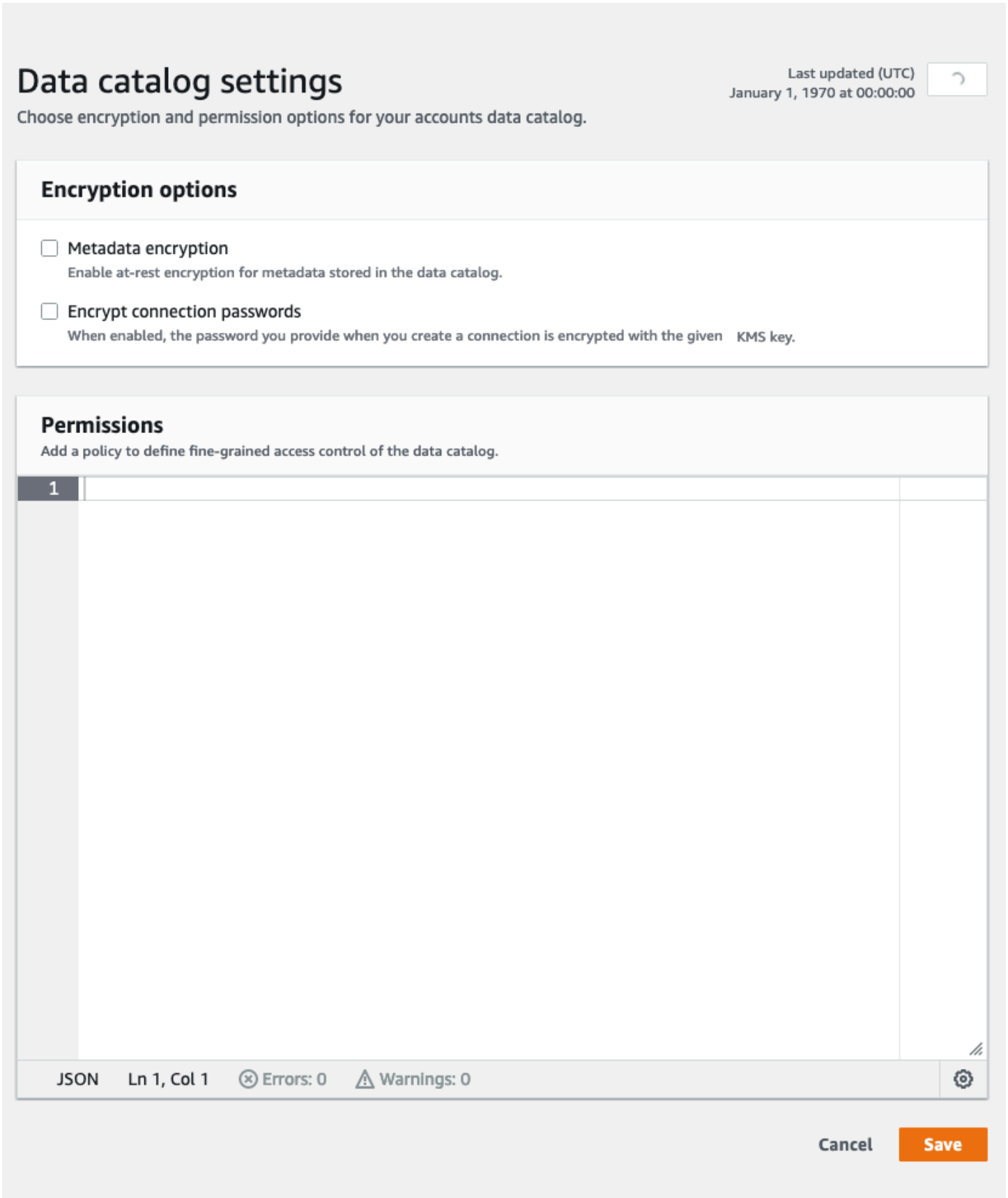
   Esta consulta crea los metadatos de la tabla en el Catálogo de datos en función del esquema y las particiones definidos sin rastrear realmente los datos.

Puede consultar la tabla en Athena, que utilizará los metadatos del Catálogo de datos para acceder a sus archivos de datos en Amazon S3 y consultarlos.

Para obtener más información, consulte Creación de bases de datos y tablas en la Guía del usuario de Amazon Athena.

Configuración del Catálogo de datos

La configuración del Catálogo de datos contiene opciones para establecer las opciones de encriptación y permisos para este en su cuenta.


Data catalog settings
Choose encryption and permission options for your accounts data catalog.
Last updated (UTC)
January 1, 1970 at 00:00:00
Encryption options
Metadata encryption
Enable at-rest encryption for metadata stored in the data catalog.
Encrypt connection passwords
When enabled, the password you provide when you create a connection is encrypted with the given KMS key.
Permissions
Add a policy to define fine-grained access control of the data catalog.
1
JSON
Ln 1, Col 1
Errors: 0
Warnings: 0
Cancel
Save

Para cambiar el control de acceso detallado del Catálogo de datos

1. Inicie sesión en la AWS Management Console y abra la consola de AWS Glue en https://console.aws.amazon.com/glue/.
2. Elija una opción de cifrado.

   - Cifrado de metadatos: seleccione esta casilla para cifrar los metadatos en el Catálogo de datos. Los metadatos se cifrarán en reposo con la clave AWS Key Management Service (AWS KMS) que especifique. Para obtener más información, consulte Cifrado del Catálogo de datos.
   - Cifrado de contraseñas de conexión: active esta casilla para cifrar las contraseñas en el objeto de conexión de AWS Glue cuando se cree o se actualice la conexión. Las contraseñas se cifran utilizando la clave de AWS KMS que especifique. Cuando se devuelven las contraseñas, se cifran. Esta opción es una configuración global para todas las conexiones de AWS Glue en el Catálogo de datos. Si desactiva esta casilla, las contraseñas cifradas anteriormente permanecen cifradas utilizando la clave que se usó al crearlas o actualizarlas. Para obtener más información acerca de las conexiones de AWS Glue, consulte Conexión a datos.

     Al habilitar esta opción, elija una clave de AWS KMS o elija Enter a key ARN (Escribir un ARN de clave) y proporcione el nombre de recurso de Amazon (ARN) de la clave. Escriba el ARN con la forma `arn:aws:kms:region:account-id:key/key-id`. También puede proporcionar el ARN como un alias de clave, como `arn:aws:kms:region:account-id:alias/alias-name`.

     > **Important**
     >
     > Si se selecciona esta opción, cualquier usuario o rol que cree o actualice una conexión debe tener permiso de `kms:Encrypt` en la clave KMS especificada.

     Para obtener más información, consulte Cifrado de las contraseñas de conexión.
3. Elija Settings (Configuración), y luego en el editor de Permissions (Permisos), agregue la instrucción de política para cambiar el control de acceso detallado del Catálogo de datos para su cuenta. Solo se puede asociar una política al Catálogo de datos a la vez. Puede pegar una política de recursos JSON en este control. Para obtener más información, consulte Políticas basadas en recursos dentro de Glue AWS.
4. Elija Save (Guardar) para actualizar el Catálogo de datos con los cambios que haya realizado.

También puede utilizar operaciones de la API de AWS Glue para colocar, obtener y eliminar políticas de recursos. Para obtener más información, consulte API de seguridad en AWS Glue.

Cómo completar y administrar las tablas transaccionales

Apache Iceberg, Apache Hudi y Linux Foundation Delta Lake son formatos de tablas de código abierto diseñados para gestionar cargas de trabajo de análisis de datos y lagos de datos a gran escala en Apache Spark.

Puede completar las tablas de Iceberg, Hudi y Delta Lake en el AWS Glue Data Catalog mediante los siguientes métodos:

- Rastreador de AWS Glue: Los Rastreador de AWS Glue pueden detectar y completar automáticamente los metadatos de las tablas de Iceberg, Hudi y Delta Lake en el Catálogo de datos. Para obtener más información, consulte Uso de rastreadores para completar el Catálogo de datos .
- Trabajos de ETL de AWS Glue: Puede crear trabajos de ETL para escribir datos en las tablas de Iceberg, Hudi y Delta Lake, y completar sus metadatos en el Catálogo de datos. Para obtener más información, consulte Uso de marcos de lagos de datos con trabajos de ETL de AWS Glue.
- Consola de AWS Glue, consola de AWS Lake Formation, AWS CLI o API: Puede usar la consola de AWS Glue, la consola de Lake Formation o API para crear y administrar las definiciones de las tablas de Iceberg en el Catálogo de datos.

Temas

- Creación de tablas de Apache Iceberg
- Optimización de las tablas de Iceberg

Creación de tablas de Apache Iceberg

Puede crear tablas de Apache Iceberg que utilicen el formato de datos de Apache Parquet en el AWS Glue Data Catalog con datos que residan en Amazon S3. Una tabla en el Catálogo de datos es la definición de metadatos que representa los datos en un almacén de datos. AWS Glue crea tablas de Iceberg v2 de forma predeterminada. Para ver la diferencia entre las tablas v1 y v2, consulte Cambios de versión de formato en la documentación de Apache Iceberg.

Apache Iceberg es un formato de tabla abierto para conjuntos de datos analíticos muy grandes. Iceberg permite modificar fácilmente su esquema, o evolución del esquema, de manera que los

usuarios pueden añadir, renombrar o eliminar columnas de una tabla de datos sin alterar los datos subyacentes. Iceberg también ofrece compatibilidad con el control de versiones de datos, que permite a los usuarios hacer un seguimiento de los cambios en los datos a lo largo del tiempo. Esto habilita la característica de viaje en el tiempo, con la que los usuarios pueden acceder a versiones históricas de los datos y consultarlas, así como analizar los cambios en los datos entre actualizaciones y eliminaciones.

Puede usar la consola de AWS Glue o de Lake Formation, o bien la operación `CreateTable` en la API de AWS Glue, para crear una tabla de Iceberg en el Catálogo de datos. Para obtener más información, consulte la acción CreateTable (Python: create_table).

Cuando cree una tabla Iceberg en el Catálogo de datos, deberá especificar el formato de la tabla y la ruta del archivo de metadatos en Amazon S3 para poder hacer lecturas y escrituras.

Puede utilizar Lake Formation para asegurar su tabla Iceberg utilizando permisos de control de acceso específicos cuando registre la ubicación de datos de Amazon S3 con AWS Lake Formation. Para los datos de origen en Amazon S3 y los metadatos que no están registrados en Lake Formation, el acceso se determina mediante las políticas de permisos de IAM para Amazon S3 y acciones de AWS Glue. Para obtener más información, consulte Administración de permisos.

> **Note**
>
> El Catálogo de datos no admite la creación de particiones ni la adición de propiedades de tablas de iceberg.

Requisitos previos

Para crear tablas Iceberg en el Catálogo de datos y configurar los permisos de acceso a los datos de Lake Formation, debe cumplir los siguientes requisitos:

1. Se requieren permisos para crear tablas de Iceberg sin datos registrados en Lake Formation.

   Además de los permisos necesarios para crear una tabla en el Catálogo de datos, el creador de la tabla requiere los siguientes permisos:

   - `s3:PutObject` en el recurso arn:aws:s3:::{bucketName}
   - `s3:GetObject` en el recurso arn:aws:s3:::{bucketName}
   - `s3:DeleteObject` en el recurso arn:aws:s3:::{bucketName}

2. Se requieren permisos para crear tablas de Iceberg con datos registrados en Lake Formation:

   Para utilizar Lake Formation para administrar y asegurar los datos de su lago de datos, registre su ubicación de Amazon S3 que tiene los datos de las tablas con Lake Formation. De este modo, Lake Formation puede suministrar credenciales a servicios analíticos de AWS como Athena, Redshift Spectrum y Amazon EMR para acceder a los datos. Para obtener más información sobre el registro de una ubicación de Amazon S3, consulte Cómo añadir una ubicación de Amazon S3 a su lago de datos.

   Una entidad principal que lee y escribe los datos subyacentes que están registrados en Lake Formation requiere los siguientes permisos:

   - `lakeformation:GetDataAccess`
   - `DATA_LOCATION_ACCESS`

     Una entidad principal que tiene permisos de localización de datos en una localización también tiene permisos de localización en todas las ubicaciones secundarias.

     Para obtener más información sobre los permisos de ubicación de datos, consulte Control de acceso a los datos subyacentes.

Para habilitar la compactación, el servicio debe asumir un rol de IAM que tenga permisos para actualizar las tablas del Catálogo de datos. Para obtener más información, consulte Requisitos previos para la optimización de tablas

Creación de tablas de Iceberg

Puede crear tablas de Iceberg v1 y v2 con la consola de AWS Glue o de Lake Formation, o bien con la AWS Command Line Interface, tal como se documenta en esta página. También puede crear tablas de Iceberg con el Rastreador de AWS Glue. Para más información, consulte Catálogo de datos y rastreadores en la Guía para desarrolladores de AWS Glue.

Para crear una tabla de Iceberg

Console

1. Inicie sesión en la AWS Management Console y abra la consola de AWS Glue en https://console.aws.amazon.com/glue/.

2. En Catálogo de datos, seleccione Tablas y utilice el botón Crear tabla para especificar los siguientes atributos:
   - Nombre de la tabla: Ingrese el nombre de la tabla. Si utiliza Athena para acceder a las tablas, utilice los consejos para nombres recogidos en la Guía del usuario de Amazon Athena.
   - Base de datos: Seleccione una base de datos existente o cree una nueva.
   - Descripción: Descripción de la tabla. Puede escribir una descripción para ayudarle a entender el contenido de la tabla.
   - Formato de la tabla: En Formato de la tabla, seleccione Apache Iceberg.
   - Activar la compactación: Seleccione Activar la compactación para compactar los objetos pequeños de Amazon S3 de la tabla de modo que formen objetos más grandes.
   - Rol de IAM: Para ejecutar la compactación, el servicio asume un rol de IAM en su nombre. Puede elegir un rol de IAM mediante el menú desplegable. Asegúrese de que el rol tenga los permisos necesarios para habilitar la compactación.

     Para obtener más información sobre los permisos necesarios, consulte Requisitos previos para la optimización de tablas .
   - Ubicación: Especifique la ruta a la carpeta en Amazon S3 que almacena la tabla de metadatos. Iceberg necesita un archivo de metadatos y una ubicación en el Catálogo de datos para poder hacer lecturas y escrituras.
   - Esquema: Seleccione Agregar columnas para añadir columnas y los tipos de datos de las columnas. Tiene la opción de crear una tabla vacía y actualizar el esquema más adelante. El Catálogo de datos admite los tipos de datos de Hive. Para obtener más información, consulte Tipos de datos de Hive.

     Con Iceberg podrá desarrollar el esquema y la partición después de crear la tabla. Puede utilizar consultas de Athena para actualizar el esquema de la tabla y consultas de Spark para actualizar las particiones.

AWS CLI

```
aws glue create-table \
    --database-name iceberg-db \
    --region us-west-2 \
    --open-table-format-input '{
      "IcebergInput": {
```

```
          "MetadataOperation": "CREATE",
          "Version": "2"
       }
    }' \
--table-input '{"Name":"test-iceberg-input-demo",
        "TableType": "EXTERNAL_TABLE",
        "StorageDescriptor":{
           "Columns":[
               {"Name":"col1", "Type":"int"},
               {"Name":"col2", "Type":"int"},
               {"Name":"col3", "Type":"string"}
            ],
           "Location":"s3://DOC_EXAMPLE_BUCKET_ICEBERG/"
        }
    }'
```

Optimización de las tablas de Iceberg

Los lagos de datos de Amazon S3 que utilizan formatos de tablas abiertas, como Apache Iceberg, almacenan los datos como objetos de Amazon S3. Tener miles de objetos pequeños de Amazon S3 en una tabla de lago de datos aumenta la sobrecarga de metadatos en las tablas Iceberg y afecta al rendimiento de lectura. Para mejorar el rendimiento de lectura de los servicios de análisis de AWS, como Amazon Athena y Amazon EMR, y los trabajos de AWS Glue ETL, AWS Glue Data Catalog proporciona una compactación administrada (un proceso que compacta objetos pequeños de Amazon S3 para convertirlos en objetos más grandes) para las tablas de Iceberg del Catálogo de datos. Puede usar la consola de AWS Glue, la consola de Lake Formation, AWS CLI, o la API de AWS para habilitar o deshabilitar la compactación de las tablas de Iceberg individuales que están en el Catálogo de datos.

El optimizador de tablas supervisa continuamente las particiones de las tablas e inicia el proceso de compactación cuando se supera el límite de cantidad y tamaño de los archivos. Las tablas de Iceberg son aptas para la compactación si el tamaño de archivo especificado en la propiedad write.target-file-size-bytes está dentro del rango de 128 MB a 512 MB. En el Catálogo de datos, el proceso de compactación comienza si la tabla tiene más de cinco archivos, cada uno inferior al 75 % de la propiedad write.target-file-size-bytes.

Por ejemplo, suponga que tiene una tabla con el límite de tamaño de archivo establecido en 512 MB en la propiedad write.target-file-size-bytes (dentro del rango prescrito de 128 MB a 512 MB) y la

tabla contiene 10 archivos. Si 6 de los 10 archivos tienen menos de 384 MB (0,75*512) cada uno, el Catálogo de datos activa la compactación.

El Catálogo de datos efectúa la compactación sin interferir con las consultas simultáneas. El catálogo de datos admite la compactación de datos solo para tablas en formato Parquet.

Para conocer los tipos de datos, los formatos de compresión y las limitaciones compatibles, consulte Formatos compatibles y limitaciones de la compactación de datos administrada .

Temas

- Requisitos previos para la optimización de tablas
- Habilitar la compactación
- Deshabilitación de la compactación
- Consultar los detalles de compactación
- Visualización de métricas de Amazon CloudWatch
- Eliminar un optimizador
- Formatos compatibles y limitaciones de la compactación de datos administrada

Requisitos previos para la optimización de tablas

El optimizador de tablas asume los permisos del rol de AWS Identity and Access Management (IAM) que especifica al habilitar la compactación de una tabla. El rol de IAM debe tener los permisos para leer los datos y actualizar los metadatos en el catálogo de datos. Cree un rol de IAM y adjunte las siguientes políticas integradas:

- Agregue la siguiente política en línea que conceda a Amazon S3 permisos de lectura y escritura en la ubicación para los datos que no estén registrados en Lake Formation. Esta política también incluye permisos para actualizar la tabla en el catálogo de datos y permitir que AWS Glue agregue registros en los registros Amazon CloudWatch y publicar métricas. Para los datos de origen en Amazon S3 que no estén registrados en Lake Formation, el acceso se determina mediante las políticas de permisos de IAM para Amazon S3 y acciones de AWS Glue.

  En las siguientes políticas en línea, sustituya `bucket-name` por el nombre de su bucket de Amazon S3, `aws-account-id` y `region` por un número de cuenta y región del catálogo de datos de AWS válidos, `database_name` por el nombre de su base de datos y `table_name` por el nombre de la tabla.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::<bucket-name>/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::<bucket-name>"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "glue:UpdateTable",
                "glue:GetTable"
            ],
            "Resource": [
                "arn:aws:glue:<region>:<aws-account-id>:table/<database-name>/<table-
name>",
                "arn:aws:glue:<region>:<aws-account-id>:database/<database-name>",
                "arn:aws:glue:<region>:<aws-account-id>:catalog"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
```

```
            "Resource": "arn:aws:logs:<region>:<aws-account-id>:log-group:/aws-glue/
iceberg-compaction/logs:*"
        }
    ]
}
```

- Utilice la siguiente política para habilitar la compactación de los datos registrados en Lake Formation.

  Si el rol de compactación no tiene permisos de grupo `IAM_ALLOWED_PRINCIPALS` otorgados en la tabla, el rol requiere los permisos de Lake Formation `ALTER`, `DESCRIBE`, `INSERT` y `DELETE` de la tabla.

  Para obtener más información sobre cómo registrar un bucket de Amazon S3 con Lake Formation, consulte Cómo añadir una ubicación de Amazon S3 a su lago de datos.

```
{
 "Version": "2012-10-17",
 "Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "lakeformation:GetDataAccess"
        ],
        "Resource": "*"
    },
    {
        "Effect": "Allow",
        "Action": [
        "glue:UpdateTable",
        "glue:GetTable"
        ],

        "Resource": [
            "arn:aws:glue:<region>:<aws-account-
id>:table/<databaseName>/<tableName>",
            "arn:aws:glue:<region>:<aws-account-id>:database/<database-name>",
            "arn:aws:glue:<region>:<aws-account-id>:catalog"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
```

```
            "logs:CreateLogGroup",
            "logs:CreateLogStream",
            "logs:PutLogEvents"
        ],
        "Resource": "arn:aws:logs:<region>:<aws-account-id>:log-group:/aws-glue/
iceberg-compaction/logs:*"
        }
    ]
 }
```

- (Opcional) Para compactar tablas Iceberg con datos de buckets de Amazon S3 cifrados mediante cifrado del lado del servidor, el rol de compactación requiere permisos para descifrar los objetos de Amazon S3 y generar una nueva clave de datos para escribir los objetos en los buckets cifrados. Agregue la siguiente política a la clave de AWS KMS deseada. Solo admitimos el cifrado a nivel de bucket.

```
{
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:aws:iam::<aws-account-id>:role/<compaction-role-name>"
    },
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
    ],
    "Resource": "*"
}
```

- (Opcional) Para la ubicación de datos registrada en Lake Formation, el rol utilizado para registrar la ubicación requiere permisos para descifrar los objetos de Amazon S3 y generar una nueva clave de datos para escribir los objetos en los buckets cifrados. Para obtener más información, consulte Registro de una ubicación de Amazon S3.
- (Opcional) Si la clave AWS KMS está almacenada en una cuenta AWS diferente, debe incluir los siguientes permisos para el rol de compactación.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": [
                "kms:Decrypt",
                "kms:GenerateDataKey"
            ],
            "Resource": ["arn:aws:kms:<REGION>:<KEY_OWNER_ACCOUNT_ID>:key/<KEY_ID>" ]
        }
    ]
}
```

- El rol que utilice para ejecutar la compactación debe tener el permiso `iam:PassRole` correspondiente al rol.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::<account-id>:role/<compaction-role-name>"
            ]
        }
    ]
}
```

- Agregue la siguiente política de confianza al rol para que el servicio AWS Glue asuma el rol de IAM para ejecutar el proceso de compactación.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": "glue.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
```

```
    ]
}
```

Habilitar la compactación

Puede usar la consola de Lake Formation, la consola de AWS Glue, AWS CLI, o la API de AWS para habilitar la compactación de las tablas de Apache Iceberg en el Catálogo de datos. Para las tablas nuevas, puede elegir Apache Iceberg como formato de tabla y habilitar la compactación al crear la tabla. La compactación está deshabilitada de forma predeterminada para las tablas nuevas.

Console

Para habilitar la compactación

1. Abra la consola de AWS Glue en https://console.aws.amazon.com/glue/ e inicie sesión como administrador del lago de datos, creador de la tabla o usuario al que se le hayan concedido los permisos `glue:UpdateTable` y `lakeformation:GetDataAccess` de la tabla.
2. En el panel de navegación, en Catálogo de datos, elija Tablas.
3. En la página Tablas, elija una tabla en formato de tabla abierta para la que desee activar la compactación y, a continuación, en el menú Acciones, seleccione Habilitar la compactación.
4. También puede activar la compactación seleccionando la tabla y abriendo la página de Detalles de la tabla. Seleccione la pestaña Optimización de tablas en la sección inferior de la página y elija Habilitar la compactación.
5. A continuación, seleccione un rol de IAM existente en el menú desplegable con los permisos que se muestran en la sección Requisitos previos para la optimización de tablas .

   Al elegir la opción Crear un nuevo rol de IAM, el servicio crea un rol personalizado con los permisos necesarios para ejecutar la compactación.

   Siga los pasos que se indican a continuación para actualizar un rol de IAM existente:

   a. Para actualizar la política de permisos del rol de IAM, en la consola de IAM, vaya al rol de IAM que se está utilizando para ejecutar la compactación.

   b. En la sección Agregar permisos, elija Crear política. En la ventana del navegador que se acaba de abrir, cree una nueva política para utilizarla con su rol.

   c. En la página Crear política, elija la pestaña JSON. Copie el código JSON que se muestra en la sección Requisitos previos en el campo del editor de políticas.

AWS CLI

En el ejemplo siguiente se muestra cómo habilitar la compactación. Sustituya el ID de cuenta por un ID de cuenta de AWS válido. Sustituya el nombre de la base de datos y el nombre de la tabla por el nombre real de la tabla Iceberg y el nombre de la base de datos. Sustituya el `roleArn` por el nombre de recurso de AWS (ARN) del rol de IAM y el nombre del rol de IAM que tiene los permisos necesarios para ejecutar la compactación.

```
aws glue create-table-optimizer \
  --catalog-id 123456789012 \
  --database-name iceberg_db \
  --table-name iceberg_table \
  --table-optimizer-configuration
 '{"roleArn":"arn:aws:iam::123456789012:role/compaction_role", "enabled":'true'}' \
  --type compaction
```

AWS API

Llame a la operación `CreateTableOptimizer` para habilitar la compactación de una tabla.

Después de activar la compactación, la pestaña de Optimización de la tabla muestra los siguientes detalles de compactación (después de aproximadamente 15 a 20 minutos):

Hora de inicio

Hora a la que se inició el proceso de compactación en Lake Formation. El valor es una marca en la hora UTC.

Hora de finalización

Hora a la que terminó el proceso de compactación en el Catálogo de datos. El valor es una marca en la hora UTC.

Status

Estado del ciclo de compactación. Los valores indican éxito o fracaso.

Archivos compactados

Número de archivos compactados.

Bytes compactados

Número de bytes compactados.

Deshabilitación de la compactación

Puede deshabilitar la compactación automática de una tabla Apache Iceberg concreta mediante la consola AWS Glue o AWS CLI.

Console

1. Elija la Base de datos y las Tablas. En la lista de tablas, elija la tabla en formato de tabla abierta en la que desee deshabilitar la compactación.
2. Puede elegir una tabla Iceberg y elegir Desactivar la compactación en Acciones.

   También puede deshabilitar la compactación de la tabla seleccionando Desactivar la compactación en la sección inferior de la página de Detalles de las tablas.
3. Selecciona Desactivar la compactación en el mensaje de confirmación. Puede volver a habilitar la compactación más adelante.

   Tras la confirmación, la compactación se desactiva y el estado de compactación de la tabla vuelve a ser el siguiente `Off`.

AWS CLI

En el siguiente ejemplo, reemplace el ID de cuenta con un ID de AWS válido. Sustituya el nombre de la base de datos y el nombre de la tabla por el nombre real de la tabla Iceberg y el nombre de la base de datos. Sustituya el `roleArn` por el nombre de recurso de AWS (ARN) del rol de IAM y el nombre real del rol de IAM que tiene los permisos necesarios para ejecutar la compactación.

```
aws glue update-table-optimizer \
  --catalog-id 123456789012 \
  --database-name iceberg_db \
  --table-name iceberg_table \
  --table-optimizer-configuration
 '{"roleArn":"arn:aws:iam::123456789012:role/compaction_role", "enabled":'false'}'\
  --type compaction
```

AWS API

Llame a la operación UpdateTableOptimizer para deshabilitar la compactación de una tabla específica.

Consultar los detalles de compactación

Puede ver el estado de compactación de Apache Iceberg mediante la consola de AWS CLI o mediante las operaciones de la API de AWS.

Console

Para ver el estado de compactación de las tablas Iceberg (consola)

- Puede ver el estado de compactación de las tablas de Iceberg en la consola de Lake Formation seleccionando Tablas en Catálogo de datos. El campo Estado de compactación muestra el estado de la operación de compactación. Puede mostrar el formato de la tabla y el estado de compactación mediante las preferencias de la tabla.
- Para ver el historial de compactación de una tabla específica, seleccione Tablas en AWS Glue Data Catalog y elija una tabla para ver los detalles de la tabla. La pestaña Optimización de la tabla muestra el historial de compactación de la tabla.

AWS CLI

Puede ver los detalles de compactación utilizando AWS CLI.

En los siguientes ejemplos, sustituya el identificador de cuenta por un identificador de cuenta de AWS válido, el nombre de la base de datos y el nombre de la tabla por el nombre real de la tabla de Iceberg.

- Para obtener los detalles de la última tanda de compactación de una tabla

```
aws get-table-optimizer \
  --catalog-id 123456789012 \
  --database-name iceberg_db \
  --table-name iceberg_table \
  --type compaction
```

- Utilice el siguiente ejemplo para recuperar el historial de un optimizador de una tabla específica.

```
aws list-table-optimizer-runs \
  --catalog-id 123456789012 \
  --database-name iceberg_db \
  --table-name iceberg_table \
  --type compaction
```

- En el siguiente ejemplo se muestra cómo recuperar la ejecución de compactación y los detalles de configuración de varios optimizadores. Puede especificar un máximo de 20 optimizadores.

```
aws glue batch-get-table-optimizer \
--entries '[{"catalogId":"123456789012", "databaseName":"iceberg_db",
 "tableName":"iceberg_table", "type":"compaction"}]'
```

AWS API

- Utilice la operación `GetTableOptimizer` para recuperar los detalles de la última ejecución de un optimizador.
- Utilice la operación `ListTableOptimizerRuns` para recuperar el historial de un optimizador determinado en una tabla específica. Puede especificar 20 optimizadores en una sola llamada a la API.
- Utilice la operación `BatchGetTableOptimizer` para recuperar los detalles de configuración de varios optimizadores de su cuenta. Esta operación no permite hacer llamadas entre cuentas.

Visualización de métricas de Amazon CloudWatch

Tras ejecutar la compactación correctamente, el servicio crea métricas Amazon CloudWatch sobre el rendimiento del trabajo de compactación. Puede ir a las Métricas de CloudWatch y elegir Métricas, Todas las métricas. Puede filtrar las métricas por el espacio de nombres específico (por ejemplo AWS Glue), el nombre de la tabla o el nombre de la base de datos.

Para obtener más información, consulte Ver métricas disponibles en la Guía del usuario de Amazon CloudWatch.

- Número de bytes compactados
- Número de archivos compactados
- Número de DPU asignadas a los trabajos

- Duración del trabajo (horas)

Eliminar un optimizador

Puede eliminar un optimizador y los metadatos asociados a la tabla mediante AWS CLI o una operación de API de AWS.

Ejecute el siguiente comando AWS CLI para eliminar el historial de compactación de una tabla.

```
aws glue delete-table-optimizer \
  --catalog-id 123456789012 \
  --database-name iceberg_db \
  --table-name iceberg_table \
  --type compaction
```

Utilice la operación `DeleteTableOptimizer` para eliminar un optimizador de una tabla.

Formatos compatibles y limitaciones de la compactación de datos administrada

Para mejorar el rendimiento de lectura de los servicios de AWS análisis, como Amazon Athena, Amazon EMR y AWS Glue ETL Jobs, AWS Glue Data Catalog proporciona compactación gestionada (un proceso que compacta pequeños objetos de Amazon S3 en objetos más grandes) para las tablas Iceberg del catálogo de datos.

La compactación de datos admite una variedad de tipos de datos y formatos de compresión para leer y escribir datos, incluida la lectura de datos de tablas cifradas.

La compactación de datos admite:

- Tipos de archivos: Parquet
- Tipos de datos: Booleano, entero, largo, flotante, doble, cadena, decimal, fecha, hora, marca temporal, cadena, identificador único universal (UUID), binario
- Compresión: Zstd, gzip, snappy, sin comprimir
- Cifrado: La compactación de datos solo admite el cifrado Amazon S3 (SSE-S3) y el cifrado KMS del lado del servidor (SSE-KMS)
- Compactación de bin pack
- Evolución del esquema

- Tablas con el tamaño de archivo objetivo (escriba. target-file-size-bytes propiedad en configuración iceberg) dentro del rango inclusivo de 128 MB a 512 MB.
- Regiones
  - Asia-Pacífico (Tokio)
  - Asia-Pacífico (Seúl)
  - Asia-Pacífico (Mumbai)
  - Asia-Pacífico (Singapur)
  - Europa (Irlanda)
  - Europa (Londres)
  - Europa (Fráncfort)
  - Este de EE. UU. (Norte de Virginia)
  - Este de EE. UU. (Ohio)
  - Oeste de EE. UU. (Norte de California)
  - América del Sur (São Paulo)
- Puede ejecutar la compactación desde la cuenta en la que reside el catálogo de datos cuando el bucket de Amazon S3 que almacena los datos subyacentes esté en otra cuenta. Para ello, el rol de compactación requiere acceso al bucket de Amazon S3.

La compactación de datos actualmente no admite:

- Tipos de archivos: Avro, ORC
- Tipos de datos: Fijos
- Compresión: Brotli, lz4
- Compactación de archivos a medida que evoluciona la especificación de la partición
- Clasificación regular o clasificación en orden Z
- Combinación o eliminación archivos: El proceso de compactación omite los archivos de datos que tienen archivos de eliminación asociados
- Compactación en tablas con varias cuentas: No se puede ejecutar la compactación en tablas con varias cuentas
- Compactación en tablas de varias regiones: No se puede ejecutar la compactación en tablas de varias regiones
- Habilitar la compactación en los enlaces de recursos

- Puntos de conexión para los buckets de Amazon S3
- Gestor de bloqueos de DynamoDB: cuando se utiliza la compactación de datos, ningún otro trabajo de carga de datos debe utilizarse como org.apache.iceberg.aws.dynamodb. `lock-impl` DynamoDbLockManager.

Administración del Catálogo de datos

El AWS Glue Data Catalog es un repositorio central de metadatos que almacena los metadatos estructurales y operativos para sus conjuntos de datos de Amazon S3. La administración eficaz del Catálogo de datos es crucial para mantener la calidad, el rendimiento, la seguridad y la gobernanza de los datos.

Si comprende y aplica estas prácticas de administración del Catálogo de datos, puede asegurarse de que sus metadatos sigan siendo precisos, eficaces y seguros, y de que estén bien gestionados a medida que el panorama de sus datos evoluciona.

En esta sección se tratan los siguientes aspectos de la administración del Catálogo de datos:

- Actualización del esquema y las particiones de la tabla: A medida que evolucionan los datos, es posible que deba actualizar el esquema de la tabla o la estructura de particiones definidos en el Catálogo de datos. Para obtener más información sobre cómo realizar estas actualizaciones mediante programación con la ETL de AWS Glue, consulte Cómo actualizar el esquema y añadir nuevas particiones al Catálogo de datos mediante trabajos de ETL de AWS Glue.
- Administración de las estadísticas de columnas: Las estadísticas de columnas precisas ayudan a optimizar los planes de las consultas y a mejorar el rendimiento. Para obtener más información sobre cómo generar, actualizar y administrar las estadísticas de columnas, consulte Cómo optimizar el rendimiento de las consultas con las estadísticas de columnas.
- Cifrado del Catálogo de datos: Para proteger los metadatos confidenciales, puede cifrar el Catálogo de datos mediante el uso de AWS Key Management Service (AWS KMS). En esta sección se explica cómo activar y administrar el cifrado del Catálogo de datos.
- Protección del Catálogo de datos con AWS Lake Formation: Lake Formation proporciona un enfoque integral para abordar la seguridad de los lagos de datos y el control del acceso a estos. Puede usar Lake Formation para proteger y controlar el acceso a su Catálogo de datos y a los datos subyacentes.

Temas

- Cómo actualizar el esquema y añadir nuevas particiones al Catálogo de datos mediante trabajos de ETL de AWS Glue
- Cómo optimizar el rendimiento de las consultas con las estadísticas de columnas
- Cifrado del Catálogo de datos
- Cómo proteger su Catálogo de datos con Lake Formation

Cómo actualizar el esquema y añadir nuevas particiones al Catálogo de datos mediante trabajos de ETL de AWS Glue

Un trabajo de extracción, transformación y carga (ETL) podría crear nuevas particiones de la tabla en el almacén de datos de destino. El esquema del conjunto de datos puede evolucionar y ser cada vez más diferente del esquema del AWS Glue Data Catalog. AWS Glue Los trabajos de ETL ahora cuentan con varias características que se pueden utilizar dentro del script de ETL para actualizar el esquema y las particiones del Data Catalog. Estas características permiten ver los resultados del trabajo de ETL en el Data Catalog sin necesidad de volver a ejecutar el rastreador.

Nuevas particiones

Si desea ver las nuevas particiones del AWS Glue Data Catalog, puede realizar una de las siguientes acciones:

- Cuando el trabajo termine, vuelva a ejecutar el rastreador y, una vez que finalice, consulte las nuevas particiones en la consola.
- Cuando el trabajo termine, podrá ver las nuevas particiones en la consola de inmediato sin necesidad de volver a ejecutar el rastreador. Puede habilitar esta característica agregando algunas líneas de código al script ETL, tal y como se muestra en los siguientes ejemplos. El código utiliza el argumento `enableUpdateCatalog` para indicar que el Data Catalog se va a actualizar durante la ejecución del trabajo a medida que se creen las nuevas particiones.

Método 1

Pase `enableUpdateCatalog` y `partitionKeys` en un argumento de opciones.

Python

```
additionalOptions = {"enableUpdateCatalog": True}
additionalOptions["partitionKeys"] = ["region", "year", "month", "day"]
```

```
sink = glueContext.write_dynamic_frame_from_catalog(frame=last_transform,
 database=<target_db_name>,

 table_name=<target_table_name>, transformation_ctx="write_sink",

 additional_options=additionalOptions)
```

Scala

```
val options = JsonOptions(Map(
    "path" -> <S3_output_path>,
    "partitionKeys" -> Seq("region", "year", "month", "day"),
    "enableUpdateCatalog" -> true))
val sink = glueContext.getCatalogSink(
    database = <target_db_name>,
    tableName = <target_table_name>,
    additionalOptions = options)sink.writeDynamicFrame(df)
```

Método 2

Pase `enableUpdateCatalog` y `partitionKeys` en `getSink()` y llame a `setCatalogInfo()` en el objeto `DataSink`.

Python

```
sink = glueContext.getSink(
    connection_type="s3",
    path="<S3_output_path>",
    enableUpdateCatalog=True,
    partitionKeys=["region", "year", "month", "day"])
sink.setFormat("json")
sink.setCatalogInfo(catalogDatabase=<target_db_name>,
 catalogTableName=<target_table_name>)
sink.writeFrame(last_transform)
```

Scala

```
val options = JsonOptions(
   Map("path" -> <S3_output_path>,
       "partitionKeys" -> Seq("region", "year", "month", "day"),
       "enableUpdateCatalog" -> true))
val sink = glueContext.getSink("s3", options).withFormat("json")
```

```
sink.setCatalogInfo(<target_db_name>, <target_table_name>)
sink.writeDynamicFrame(df)
```

Ahora, puede crear nuevas tablas de catálogo, actualizar las tablas existentes con un esquema modificado y agregar nuevas particiones de tabla en el Data Catalog utilizando únicamente un trabajo de ETL de AWS Glue, sin necesidad de volver a ejecutar los rastreadores.

Actualización del esquema de la tabla

Si desea sobrescribir el esquema de la tabla del Data Catalog, puede realizar una de las siguientes acciones:

- Cuando termine el trabajo, vuelva a ejecutar el rastreador y asegúrese de que esté configurado para actualizar también la definición de la tabla. Cuando el rastreador finalice, consulte las nuevas particiones en la consola junto con las actualizaciones del esquema. Para obtener más información, consulte Configuración de un rastreador con la API.
- Cuando el trabajo termine, podrá ver de inmediato el esquema modificado en la consola sin necesidad de volver a ejecutar el rastreador. Puede habilitar esta característica agregando algunas líneas de código al script ETL, tal y como se muestra en los siguientes ejemplos. El código utiliza el valor `enableUpdateCatalog` establecido en verdadero y también el valor `updateBehavior` establecido en `UPDATE_IN_DATABASE`, lo que indica que el esquema debe sobrescribirse y deben agregarse nuevas particiones en el Data Catalog durante la ejecución del trabajo.

Python

```
additionalOptions = {
    "enableUpdateCatalog": True,
    "updateBehavior": "UPDATE_IN_DATABASE"}
additionalOptions["partitionKeys"] = ["partition_key0", "partition_key1"]

sink = glueContext.write_dynamic_frame_from_catalog(frame=last_transform,
 database=<dst_db_name>,
    table_name=<dst_tbl_name>, transformation_ctx="write_sink",
    additional_options=additionalOptions)
job.commit()
```

Scala

```
val options = JsonOptions(Map(
```

```
    "path" -> outputPath,
    "partitionKeys" -> Seq("partition_0", "partition_1"),
    "enableUpdateCatalog" -> true))
val sink = glueContext.getCatalogSink(database = nameSpace, tableName = tableName,
 additionalOptions = options)
sink.writeDynamicFrame(df)
```

También puede establecer el valor `updateBehavior` en `LOG` si no desea que el esquema de la tabla se sobrescriba pero sí quiere agregar las nuevas particiones. El valor predeterminado de `updateBehavior` es `UPDATE_IN_DATABASE`, por lo que, si no lo define explícitamente, se sobrescribirá el esquema de la tabla.

Si `enableUpdateCatalog` no se establece en verdadero, independientemente de la opción seleccionada en `updateBehavior`, el trabajo de ETL no actualizará la tabla del Data Catalog.

Creación de nuevas tablas

También puede utilizar las mismas opciones para crear una nueva tabla en el Data Catalog. Puede especificar la base de datos y el nuevo nombre de la tabla con `setCatalogInfo`.

Python

```
sink = glueContext.getSink(connection_type="s3", path="s3://path/to/data",
    enableUpdateCatalog=True, updateBehavior="UPDATE_IN_DATABASE",
    partitionKeys=["partition_key0", "partition_key1"])
sink.setFormat("<format>")
sink.setCatalogInfo(catalogDatabase=<dst_db_name>, catalogTableName=<dst_tbl_name>)
sink.writeFrame(last_transform)
```

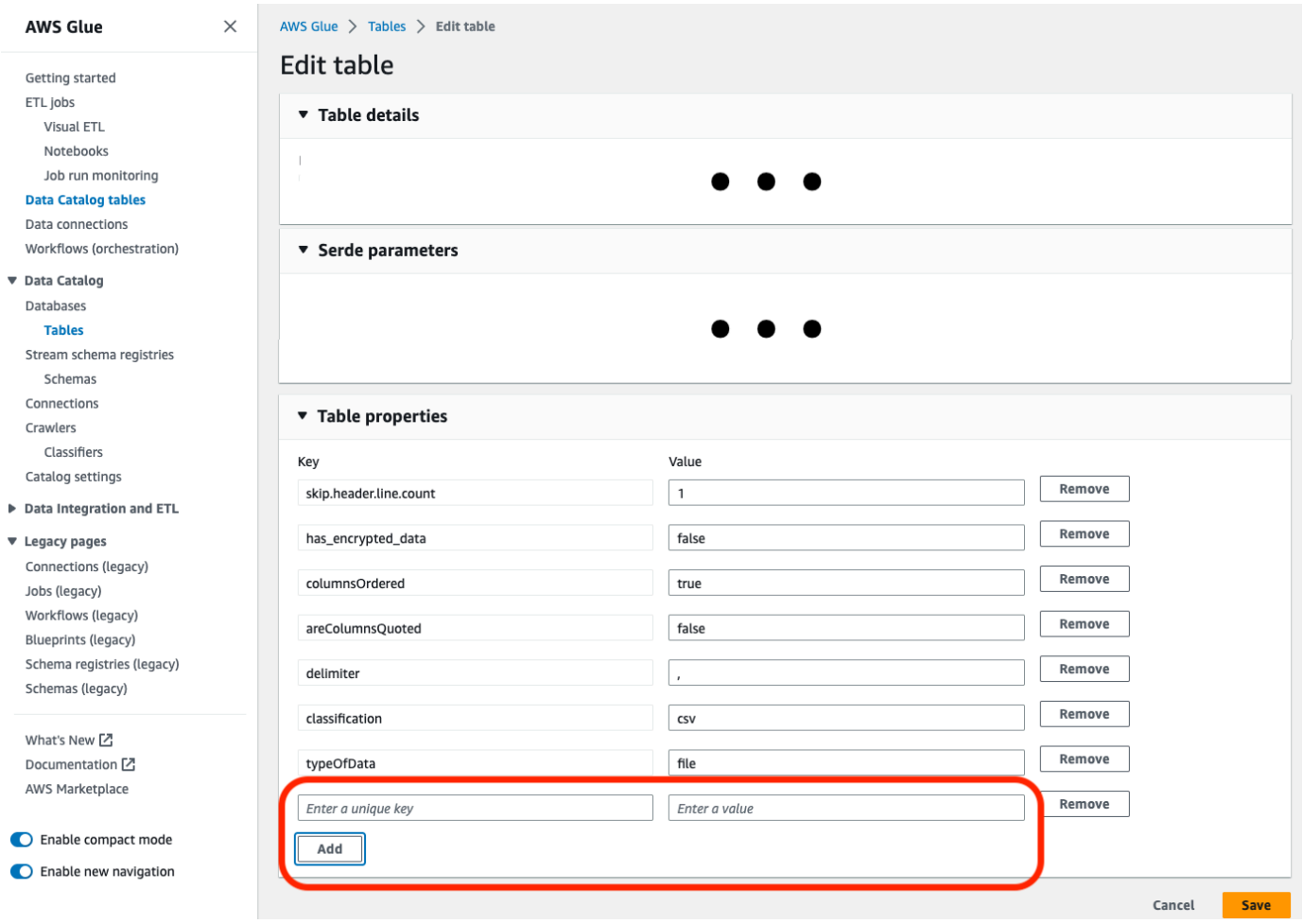
Scala

```
val options = JsonOptions(Map(
    "path" -> outputPath,
    "partitionKeys" -> Seq("<partition_1>", "<partition_2>"),
    "enableUpdateCatalog" -> true,
    "updateBehavior" -> "UPDATE_IN_DATABASE"))
val sink = glueContext.getSink(connectionType = "s3", connectionOptions =
 options).withFormat("<format>")
sink.setCatalogInfo(catalogDatabase = “<dst_db_name>”, catalogTableName =
 “<dst_tbl_name>”)
sink.writeDynamicFrame(df)
```

Restricciones

Tome nota de las siguientes restricciones:

- Solo se soportan los destinos de Amazon Simple Storage Service (Amazon S3).
- La característica `enableUpdateCatalog` no es compatible con las tablas gobernadas.
- Solo se admiten los siguientes formatos: `json`, `csv`, `avro` y `parquet`.
- Para crear o actualiza tablas con la clasificación `parquet`, debe utilizar el escritor de parquet optimizado para DynamicFrames de AWS Glue. Esto se puede lograr con uno de los siguientes:
  - Si va a actualizar una tabla existente en el catálogo con una clasificación `parquet`, la tabla debe tener la propiedad de tabla `"useGlueParquetWriter"` establecida en `true` antes de actualizarla. Puede configurar esta propiedad a través de las API/SDK de AWS Glue, la consola o una instrucción DDL de Athena.



Una vez establecida la propiedad de la tabla de catálogo, puede utilizar el siguiente fragmento de código para actualizar la tabla de catálogo con los nuevos datos:

```
glueContext.write_dynamic_frame.from_catalog(
    frame=frameToWrite,
    database="dbName",
    table_name="tableName",
    additional_options={
        "enableUpdateCatalog": True,
        "updateBehavior": "UPDATE_IN_DATABASE"
    }
)
```

- Si la tabla aún no existe en el catálogo, puede utilizar el método `getSink()` del script con `connection_type="s3"` para agregar la tabla y sus particiones al catálogo, además de escribir los datos en Amazon S3. Proporcione la `partitionKeys` adecuada y la `compression` para su flujo de trabajo.

```
s3sink = glueContext.getSink(
    path="s3://bucket/folder/",
    connection_type="s3",
    updateBehavior="UPDATE_IN_DATABASE",
    partitionKeys=[],
    compression="snappy",
    enableUpdateCatalog=True
)

s3sink.setCatalogInfo(
    catalogDatabase="dbName", catalogTableName="tableName"
)

s3sink.setFormat("parquet", useGlueParquetWriter=true)
s3sink.writeFrame(frameToWrite)
```

  - El valor de formato `glueparquet` es un método heredado para habilitar el escritor de Parquet de AWS Glue.

- Si `updateBehavior` se configura en `LOG`, las nuevas particiones solo se agregarán si el esquema de `DynamicFrame` es igual al subconjunto de las columnas definidas en el esquema de la tabla del Data Catalog, o si contiene un subconjunto de este tipo.
- Las actualizaciones de esquema no son compatibles con las tablas sin particiones (no se utiliza la opción "PartitionKeys").

- Las claves de partición deben ser equivalentes y estar en el mismo orden en el parámetro que se transfirió en el script de ETL y en las claves de partición del esquema de la tabla del catálogo de datos.
- Esta característica aún no admite la actualización/creación de tablas en las que se anidan los esquemas de actualización (por ejemplo, matrices dentro de estructuras).

Para obtener más información, consulte the section called “Glue para Spark AWS”.

Trabajar con conexiones MongoDB en trabajos de ETL

Puede crear una conexión para MongoDB y luego usar esa conexión en su trabajo de AWS Glue. Para obtener más información, consulte the section called “Conexiones a MongoDB” en la guía de progrmación de AWS Glue. La `url`, el `username` y la `password` de la conexión se almacenan en la conexión de MongoDB. Se pueden especificar otras opciones en el script de trabajo de ETL mediante el parámetro `additionalOptions` de `glueContext.getCatalogSource`. Las otras opciones pueden incluir lo siguiente:

- `database`: (obligatorio) la base de datos de MongoDB de la que se va a leer.
- `collection`: (obligatorio) la colección de MongoDB de la que se va a leer.

Al colocar la información de `database` y `collection` dentro del script de trabajo de ETL, puede utilizar la misma conexión para varios trabajos.

1. Creación de una conexión de AWS Glue Data Catalog para el origen de datos MongoDB. Consulte "connectionType": "mongodb" para ver una descripción de los parámetros de conexión. Puede crear la conexión mediante la consola, las API o CLI.
2. Cree una base de datos en AWS Glue Data Catalog para almacenar las definiciones de la tabla para sus datos de MongoDB. Para obtener más información, consulte Creación de bases de datos.
3. Cree un rastreador que rastree los datos en MongoDB usando la información en la conexión para conectarse a MongoDB. El rastreador crea las tablas en AWS Glue Data Catalog, que describen las tablas en la base de datos MongoDB que usa en su trabajo. Para obtener más información, consulte Uso de rastreadores para completar el Catálogo de datos .
4. Cree un trabajo con un script personalizado. Puede crear un trabajo mediante la consola, la API o CLI. Para obtener más información, consulte Agregar trabajos en AWS Glue.

5. Elija los destinos de datos para su trabajo. Las tablas que representan el destino de datos se pueden definir en el Data Catalog, o bien, el trabajo puede crear las tablas de destino cuando se ejecute. El usuario elige la ubicación de destino al crear el trabajo. Si el destino requiere una conexión, esta también estará referenciada en el trabajo. Si el trabajo requiere varios destinos de datos, puede agregarlos más tarde al editar el script.
6. Personalice el entorno de procesamiento del flujo de trabajo mediante argumentos para el flujo de trabajo y el script que se generó.

   A continuación, se muestra un ejemplo de creación de un `DynamicFrame` desde la base de datos MongoDB en función de la estructura de la tabla definida en el Data Catalog. El código utiliza `additionalOptions` para proporcionar la información adicional del origen de datos:

   Scala

   ```
   val resultFrame: DynamicFrame = glueContext.getCatalogSource(
           database = catalogDB,
           tableName = catalogTable,
           additionalOptions = JsonOptions(Map("database" -> DATABASE_NAME,
                   "collection" -> COLLECTION_NAME))
         ).getDynamicFrame()
   ```

   Python

   ```
   glue_context.create_dynamic_frame_from_catalog(
           database = catalogDB,
           table_name = catalogTable,
           additional_options = {"database":"database_name",
               "collection":"collection_name"})
   ```

7. Ejecute el trabajo, ya sea bajo demanda o a través de un desencadenador.

Cómo optimizar el rendimiento de las consultas con las estadísticas de columnas

Puede calcular estadísticas a nivel de columna para tablas AWS Glue Data Catalog en formatos de datos como Parquet, ORC, JSON, ION, CSV y XML sin necesidad de configurar canalizaciones de datos adicionales. Las estadísticas de columnas le ayudan a entender los perfiles de datos al obtener información sobre los valores de una columna. El catálogo de datos permite generar estadísticas para los valores de las columnas, como el valor mínimo, el valor máximo, los valores nulos totales,

los valores distintos totales, la longitud media de los valores y el total de apariciones de valores verdaderos.

Los servicios analíticos AWS como Amazon Redshift y Amazon Athena pueden utilizar estas estadísticas de columnas para generar planes de ejecución de consultas y elegir el plan óptimo que mejore el rendimiento de las consultas.

Puede configurarlo para ejecutar la tarea de generación de estadísticas de columnas mediante la consola AWS Glue o AWS CLI. Al iniciar el proceso, AWS Glue inicia un trabajo de Spark en segundo plano y actualiza los metadatos de la tabla AWS Glue en el catálogo de datos. Puede ver las estadísticas de las columnas mediante la consola AWS Glue o AWS CLI, o llamando a la operación de la API GetColumnStatisticsForTable.

> **Note**
>
> Si utiliza los permisos de Lake Formation para controlar el acceso a la tabla, el rol que asume la tarea de estadísticas de columnas requiere acceso total a la tabla para generar estadísticas.

Temas

- Requisitos previos para generar estadísticas de columnas
- Generación de estadísticas de columnas
- Visualización de estadísticas de columnas
- Actualización de estadísticas de columnas
- Eliminar estadísticas de columnas
- Visualización de las ejecuciones de tareas de estadísticas de columnas
- Detener la ejecución de la tarea de estadísticas de columnas
- Consideraciones y limitaciones

Requisitos previos para generar estadísticas de columnas

Para generar o actualizar las estadísticas de las columnas, la tarea de generación de estadísticas adopta un rol de AWS Identity and Access Management (IAM) en su nombre. Según los permisos concedidos al rol, la tarea de generación de estadísticas de columnas puede leer los datos del almacén de datos de Amazon S3.

> **Note**
>
> Para generar estadísticas para las tablas administradas por Lake Formation, el rol de IAM utilizado para generar estadísticas requiere acceso total a las tablas.

Para utilizar el control de acceso basado en roles, debe crear un rol de IAM con los permisos que se indican en la política a continuación y debe agregarlo a la tarea de generación de estadísticas de la columna.

Para crear un rol de IAM para generar estadísticas de columnas

1. Para crear un rol de IAM;, consulte Crear un rol de IAM para AWS Glue.
2. Para actualizar un rol existente, en la consola de IAM, vaya al rol de IAM que está utilizando el proceso de generación de estadísticas de columnas.
3. En la sección Agregar permisos, elija Asociar políticas. En la ventana del navegador que se acaba de abrir, elija la política administrada `AWSGlueServiceRole` AWS.
4. También debe incluir los permisos de lectura de los datos de la ubicación de datos de Amazon S3.

   En la sección Agregar permisos, elija Crear política. En la ventana del navegador que se acaba de abrir, cree una nueva política para utilizarla con su rol.
5. En la página Crear política, elija la pestaña JSON. Copie el siguiente código `JSON` en el campo del editor de política.

   > **Note**
   >
   > En las políticas a continuación, reemplace la ID de cuenta con una Cuenta de AWS válida, `region` con la Región de la tabla y `bucket-name` con el nombre del bucket de Amazon S3.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3BucketAccess",
            "Effect": "Allow",
```

```
            "Action": [
                "s3:ListBucket",
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::<bucket-name>/*",
                "arn:aws:s3:::<bucket-name>"
            ]
        }
     ]
}
```

6. (Opcional) Si utiliza los permisos de Lake Formation para acceder a sus datos, el rol de IAM requiere permisos `lakeformation:GetDataAccess`.

```
{
    "Version": "2012-10-17",
    "Statement": [
       {
            "Sid": "LakeFormationDataAccess",
            "Effect": "Allow",
            "Action": "lakeformation:GetDataAccess",
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Si la ubicación de datos de Amazon S3 está registrada en Lake Formation y el rol de IAM que asume la tarea de generación de estadísticas de columnas no tiene permisos de grupos `IAM_ALLOWED_PRINCIPALS` concedidos en la tabla, el rol requiere los permisos de Lake Formation `ALTER` y `DESCRIBE` en la tabla. El rol utilizado para registrar el bucket de Amazon S3 requiere los permisos de Lake Formation `INSERT` y `DELETE` en la tabla.

Si la ubicación de datos de Amazon S3 no está registrada en Lake Formation y el rol de IAM no tiene permisos de grupo `IAM_ALLOWED_PRINCIPALS` otorgados en la tabla, el rol requiere los permisos de Lake Formation `ALTER`, `DESCRIBE`, `INSERT` y `DELETE` en la tabla.

7. (Opcional) La tarea de generación de estadísticas de columnas que escribe Amazon CloudWatch Logs cifrado necesita los siguientes permisos en la política de claves.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Sid": "CWLogsKmsPermissions",
        "Effect":  "Allow",
        "Action": [
            "logs:CreateLogGroup",
            "logs:CreateLogStream",
            "logs:PutLogEvents",
            "logs:AssociateKmsKey"
        ],
        "Resource": [
            "arn:aws:logs:<region>:111122223333:log-group:/aws-glue:*"
        ]
    },
    {
        "Sid": "KmsPermissions",
        "Effect": "Allow",
        "Action": [
            "kms:GenerateDataKey",
            "kms:Decrypt",
            "kms:Encrypt"
        ],
        "Resource": [
        "arn:aws:kms:<region>:111122223333:key/"arn of key used for ETL cloudwatch
 encryption"
        ],
        "Condition": {
            "StringEquals": {
                "kms:ViaService": ["glue.<region>.amazonaws.com"]
            }
        }
    }
    ]
}
```

8. El rol que utilice para ejecutar las estadística de columnas debe contener el permiso `iam:PassRole` en el rol.

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "iam:PassRole"
        ],
        "Resource": [
            "arn:aws:iam::111122223333:role/<columnstats-role-name>"
        ]
    }]
}
```

9. Al crear un rol de IAM para generar estadísticas de columnas, el rol también debe tener la siguiente política de confianza que permita al servicio asumirlo.

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Sid": "TrustPolicy",
        "Effect": "Allow",
        "Principal": {
          "Service": "glue.amazonaws.com"
        },
        "Action": "sts:AssumeRole",
      }
   ]
}
```

Generación de estadísticas de columnas

Siga estos pasos para administrar la generación de estadísticas en el catálogo de datos mediante la consola AWS Glue o AWS CLI.

Console

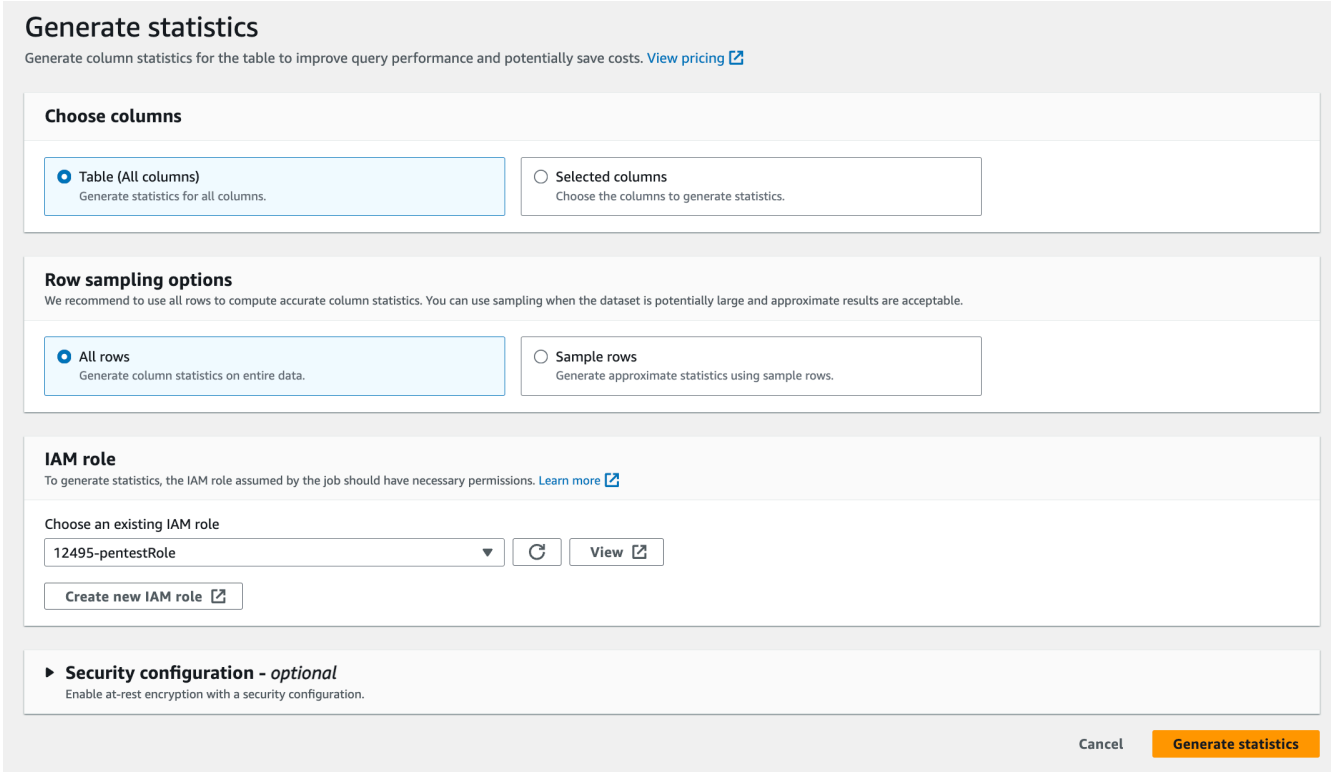
Para generar estadísticas de columnas mediante la consola

1. Inicie sesión en la consola de AWS Glue en https://console.aws.amazon.com/glue/.
2. Seleccione las tablas del Data Catalog.
3. Seleccione una tabla de la lista.

4. Seleccione Generar estadísticas en el menú Acciones.

   También puede pulsar el botón Generar estadísticas en la pestaña Estadísticas de columnas, en la sección inferior de la página de Tablas.

5. En la página Generar estadísticas, especifique las siguientes opciones:



- Tabla (todas las columnas): elija esta opción para generar estadísticas para todas las columnas de la tabla.
- Columnas seleccionadas: elija esta opción para generar estadísticas para columnas específicas. Puede seleccionar las columnas en la lista desplegable.
- Todas las filas: elija todas las filas de la tabla para generar estadísticas precisas.
- Filas de muestra: elija solo un porcentaje específico de filas de la tabla para generar estadísticas. El valor predeterminado es Todas las filas. Utilice las flechas hacia arriba y hacia abajo para aumentar o disminuir el valor porcentual.

> **Note**
>
> Se recomienda incluir todas las filas de la tabla para calcular estadísticas precisas. Utilice filas de muestra para generar estadísticas de columnas solo cuando los valores aproximados sean aceptables.

6. (Opcional) A continuación, elija una configuración de seguridad para habilitar el cifrado en reposo de los registros.
7. Elija Generar estadísticas para ejecutar el proceso.

AWS CLI

En el siguiente ejemplo, sustituya los valores de `DatabaseName`, `TableName` y `ColumnNameList` con nombres reales de bases de datos, tablas y columnas. Sustituya el identificador de cuenta por una Cuenta de AWS válida y el nombre del rol por el nombre del rol de IAM que está utilizando para generar estadísticas.

```
aws glue start-column-statistics-task-run --input-cli-json file://input.json
{
    "DatabaseName": "<test-db>",
    "TableName": "<test-table>",
    "ColumnNameList": [
        "<column1>",
        "<column2>",
    ],
    "Role": "arn:aws:iam::<123456789012>:role/<Stats-Role>",
    "SampleSize": 10.0
}
```

También puede generar estadísticas de columnas llamando a la operación StartColumnStatisticsTaskRun.

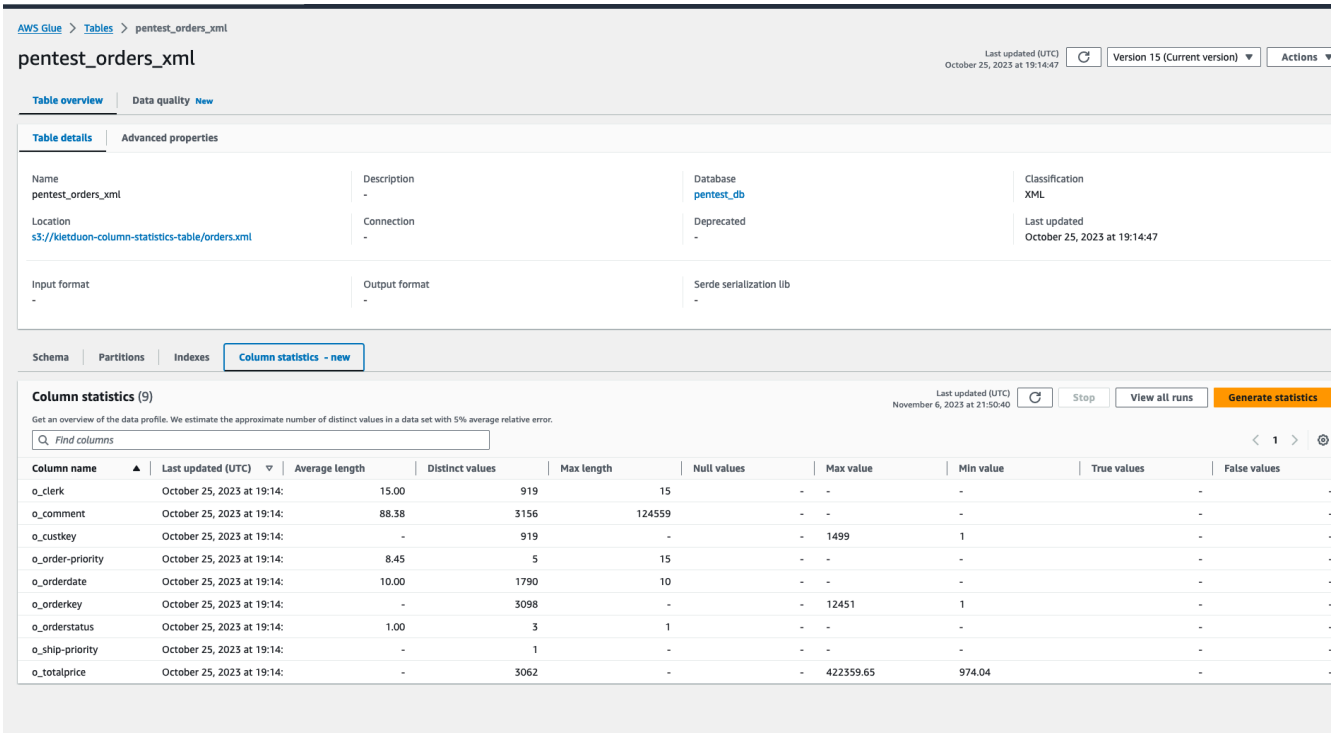
Visualización de estadísticas de columnas

Tras generar las estadísticas correctamente, el catálogo de datos almacena esta información para que los optimizadores basados en los costos de Amazon Athena y Amazon Redshift puedan tomar las decisiones óptimas al ejecutar consultas. Las estadísticas varían en función del tipo de columna.

AWS Management Console

Para ver las estadísticas de columna de una tabla

- Tras ejecutar la tarea de estadísticas de columnas, la pestaña Estadísticas de columnas de la página de Detalles de la tabla muestra las estadísticas de la tabla.



Están disponibles las siguientes estadísticas:

- Nombre de columna: nombre de columna utilizado para generar estadísticas
- Última actualización: fecha y hora en que se generaron las estadísticas
- Longitud media: longitud media de los valores de la columna
- Valores distintos: número total de valores distintos de la columna. Estimamos el número de valores distintos de una columna con un error relativo del 5 %.
- Valor máximo: el valor más alto de la columna.
- Valor mínimo: el valor más bajo de la columna.
- Longitud máxima: longitud del valor más alto de la columna.
- Valores nulos: el número total de valores nulos en la columna.
- Valores verdaderos: el número de valores verdaderos en la columna.
- Valores falsos: el número de valores falsos en la columna.

AWS CLI

En el siguiente ejemplo se muestra cómo recuperar estadísticas de columnas mediante AWS CLI.

```
aws glue get-column-statistics-for-table \
    --database-name <test_db> \
    --table-name <test_tble> \
    --column-names <col1>
```

También puede ver las estadísticas de las columnas mediante la operación de API GetColumnStatisticsForTable.

Actualización de estadísticas de columnas

Mantener las estadísticas actualizadas mejora el rendimiento de las consultas, ya que permite que el planificador de consultas elija los planes óptimos. Debe ejecutar explícitamente la tarea Generar estadísticas desde la consola AWS Glue para actualizar las estadísticas de las columnas. El catálogo de datos no actualiza automáticamente las estadísticas.

Si no utiliza la característica de generación de estadísticas de AWS Glue en la consola, puede actualizar manualmente las estadísticas de las columnas mediante la operación de API UpdateColumnStatisticsForTable o AWS CLI. En el siguiente ejemplo, se muestra cómo actualizar las estadísticas de las columnas mediante AWS CLI.

```
aws glue update-column-statistics-for-table --cli-input-json:

{
    "CatalogId": "111122223333",
    "DatabaseName": "test_db",
    "TableName": "test_table",
    "ColumnStatisticsList": [
        {
            "ColumnName": "col1",
            "ColumnType": "Boolean",
            "AnalyzedTime": "1970-01-01T00:00:00",
            "StatisticsData": {
                "Type": "BOOLEAN",
                "BooleanColumnStatisticsData": {
                    "NumberOfTrues": 5,
                    "NumberOfFalses": 5,
```

```
                    "NumberOfNulls": 0
                }
            }
        }
    ]
}
```

Eliminar estadísticas de columnas

Puede eliminar las estadísticas de las columnas mediante la operación de API DeleteColumnStatisticsForTable o AWS CLI. En el ejemplo siguiente se muestra cómo eliminar estadísticas de columnas mediante AWS Command Line Interface (AWS CLI).

```
aws glue delete-column-statistics-for-table \
    --database-name test_db \
    --table-name test_table \
    --column-name col1
```

Visualización de las ejecuciones de tareas de estadísticas de columnas

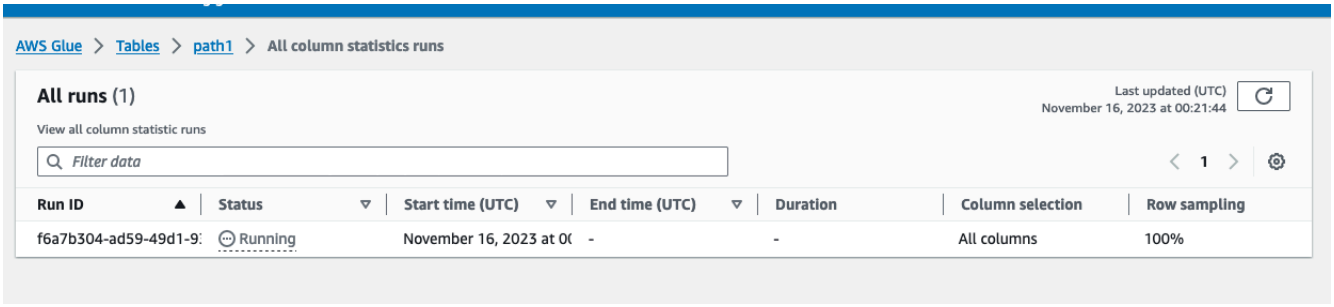
Después de ejecutar una tarea de estadísticas de columnas, puede explorar los detalles de la ejecución de la tarea de una tabla mediante la consola AWS Glue, AWS CLI o mediante la operación GetColumnStatisticsTaskRuns.

Console

Para ver los detalles de ejecución de la tarea con estadísticas de columnas

1. En la consola AWS Glue, seleccione Tablas en el catálogo de datos.
2. Seleccione una tabla con estadísticas de columnas.
3. En la página de Detalles de la tabla, seleccione Estadísticas de columnas.
4. Seleccione Ver ejecuciones.

   Puede ver información sobre todas las ejecuciones asociadas a la tabla especificada.



AWS CLI

En el siguiente ejemplo, sustituya los valores de `DatabaseName` y `TableName` por el nombre real de la base de datos y la tabla.

```
aws glue get-column-statistics-task-runs --input-cli-json file://input.json
{
    "DatabaseName": "<test-db>",
    "TableName": "<test-table>"
}
```

Detener la ejecución de la tarea de estadísticas de columnas

Puede detener la ejecución de una tarea de estadísticas de columnas para una tabla mediante la consola AWS Glue, AWS CLI o mediante la operación StopColumnStatisticsTaskRun.

Console

Para detener una tarea de estadísticas de columnas, ejecute

1. En la consola AWS Glue, seleccione Tablas en el catálogo de datos.
2. Seleccione la tabla con la columna de La ejecución de la tarea de estadísticas está en curso.
3. En la página de Detalles de la tabla, seleccione Estadísticas de columnas.
4. Elija Detener.

   Si detiene la tarea antes de que se complete la ejecución, no se generarán las estadísticas de las columnas para la tabla.

AWS CLI

En el siguiente ejemplo, sustituya los valores de `DatabaseName` y `TableName` por el nombre real de la base de datos y la tabla.

```
aws glue stop-column-statistics-task-run --input-cli-json file://input.json
{
    "DatabaseName": "<test-db>",
    "TableName": "<test-table>"
}
```

Consideraciones y limitaciones

Las siguientes consideraciones y limitaciones se aplican a la generación de estadísticas de columnas.

Consideraciones

- El uso del muestreo para generar estadísticas reduce el tiempo de ejecución, pero puede generar estadísticas inexactas.
- La ejecución de las estadísticas de cada columna requiere procesar todo el conjunto de datos.
- El catálogo de datos no almacena versiones diferentes de las estadísticas.
- Solo puede ejecutar una tarea de generación de estadísticas a la vez por tabla.
- Si una tabla se cifra con la clave AWS KMS de cliente registrada en el catálogo de datos, AWS Glue utiliza la misma clave para cifrar las estadísticas.

La tarea de estadísticas de las columnas permite generar estadísticas:

- Cuando el rol de IAM tiene permisos de tabla completos (IAM o Lake Formation).
- Cuando el rol de IAM tiene permisos sobre la tabla mediante el modo de acceso híbrido de Lake Formation.

La tarea de estadísticas de columnas no admite la generación de estadísticas para:

- Mesas con control de acceso basado en celdas de Lake Formation.
- Lagos de datos transaccionales: Fundación Linux Delta Lake, Apache Iceberg, Apache Hudi.

- Tablas en bases de datos federadas: recursos compartidos de datos Hive metastore, Amazon Redshift
- Columnas anidadas, matrices y tipos de datos de estructura.
- Tabla que se comparte con usted desde otra cuenta.

Cifrado del Catálogo de datos

Para proteger los metadatos en reposo que están almacenados en el AWS Glue Data Catalog, puede usar claves de cifrado administradas por AWS Key Management Service (AWS KMS). En Configuración del Catálogo de datos, puede activar el cifrado del Catálogo de datos para el nuevo catálogo. Puede activar o desactivar el cifrado del Catálogo de datos existente, según lo que necesite. Cuando está activado, AWS Glue cifra todos los metadatos nuevos que se escriben en el catálogo, mientras que los ya existentes permanecen sin cifrar.

Para obtener información detallada sobre el cifrado del Catálogo de datos, consulte Cifrado del Catálogo de datos.

Cómo proteger su Catálogo de datos con Lake Formation

AWS Lake Formation es un servicio que facilita la configuración de un lago de datos seguro en AWS. Proporciona un lugar central para crear y administrar sus lagos de datos de forma segura mediante la definición de permisos detallados para controlar el acceso. Lake Formation usa el Catálogo de datos para almacenar y recuperar metadatos sobre su lago de datos, como las definiciones de tablas, la información de los esquemas y las configuraciones del control de acceso a los datos.

Puede registrar la ubicación de sus datos de Amazon S3 de la base de datos o la tabla de metadatos con Lake Formation y usarla para definir los permisos a nivel de metadatos en los recursos del Catálogo de datos. También puede usar Lake Formation para administrar los permisos de acceso al almacenamiento en los datos subyacentes que se guardan en Amazon S3 en nombre de los motores analíticos integrados.

Para obtener más información, consulte ¿Qué es AWS Lake Formation?.

Cómo acceder al Catálogo de datos

Puede usar el AWS Glue Data Catalog para detectar y comprender sus datos. El Catálogo de datos proporciona una forma coherente de mantener las definiciones de los esquemas, los tipos de datos,

las ubicaciones y otros metadatos. Puede acceder al Catálogo de datos mediante los siguientes métodos:

- Consola de AWS Glue: Puede acceder al Catálogo de datos y administrarlo por medio de la consola de AWS Glue, una interfaz de usuario en línea. La consola le permite explorar y buscar bases de datos, tablas y sus metadatos asociados, así como crear, actualizar y eliminar definiciones de metadatos.
- Rastreador de AWS Glue: Los rastreadores son programas que escanean automáticamente los orígenes de datos y completan el Catálogo de datos con metadatos. Puede crear y ejecutar rastreadores para detectar y catalogar datos de diversas fuentes, como Amazon S3, Amazon RDS, Amazon DynamoDB, Amazon CloudWatch y bases de datos relacionales que cumplen los requisitos de JDBC, como MySQL y PostgreSQL, así como de varias fuentes ajenas a AWS, como Snowflake y Google BigQuery.
- API de AWS Glue: Puede acceder al Catálogo de datos por medio de programación mediante las API de AWS Glue. Estas API le permiten interactuar con el Catálogo de datos por medio de programación, lo que permite la automatización y la integración con otras aplicaciones y servicios.
- AWS Command Line Interface (AWS CLI): Puede usar la AWS CLI para acceder al Catálogo de datos y administrarlo desde la línea de comandos. La CLI proporciona comandos para crear, actualizar y eliminar definiciones de metadatos, así como para consultar y recuperar información de metadatos.
- Integración con otros servicios de AWS: El Catálogo de datos se integra con otros servicios de AWS, lo que le permite acceder a los metadatos almacenados en el Catálogo y usarlos. Por ejemplo, puede usar Amazon Athena para consultar orígenes de datos mediante los metadatos del Catálogo de datos y usar AWS Lake Formation para administrar el acceso a los datos y la gobernanza de los recursos del Catálogo.

Prácticas recomendadas para el uso del Catálogo de datos de AWS Glue

En esta sección se describen las prácticas recomendadas para administrar y usar el AWS Glue Data Catalog de forma eficaz. Se hace hincapié en prácticas como el uso eficiente de los rastreadores, la organización de los metadatos, la seguridad, la optimización del rendimiento, la automatización, la gobernanza de los datos y la integración con otros servicios de AWS.

- Use los rastreadores de forma eficaz: Ejecute los rastreadores con regularidad para mantener el Catálogo de datos actualizado con los cambios en los orígenes de datos. Use rastreos progresivos para los orígenes de datos que cambian con frecuencia a fin de mejorar el rendimiento. Configure los rastreadores para añadir automáticamente nuevas particiones o actualizar los esquemas cuando se detecten cambios.
- Organice y asigne nombres a las tablas de metadatos: Establezca una convención de nomenclatura coherente para las bases de datos y las tablas del Catálogo de datos. Agrupe los orígenes de datos relacionados en bases de datos o carpetas de forma lógica para una mejor organización. Use nombres descriptivos que expresen el propósito y el contenido de cada tabla.
- Administre los esquemas de forma eficaz: Aproveche las capacidades de inferencia de los esquemas que tienen los rastreadores de AWS Glue. Revise y actualice los cambios de esquema antes de aplicarlos para evitar interrumpir las aplicaciones posteriores. Use las características de evolución del esquema para gestionar los cambios de este con fluidez.
- Proteja el Catálogo de datos: Active el cifrado de datos en reposo y en tránsito para el Catálogo de datos. Implemente políticas de control de acceso detalladas para restringir el acceso a los datos confidenciales. Audite y revise periódicamente los permisos y los registros de actividad del Catálogo de datos.
- Integre otros servicios de AWS: Use el Catálogo de datos como una capa de metadatos centralizada para servicios como Amazon Athena, Redshift Spectrum y AWS Lake Formation. Aproveche los trabajos de ETL de AWS Glue para transformar y cargar datos en varios almacenes de datos y, al mismo tiempo, mantener los metadatos en el Catálogo de datos.
- Supervise y optimice el rendimiento: Supervise el rendimiento de los rastreadores y los trabajos de ETL mediante el uso de las métricas de Amazon CloudWatch. Particione los conjuntos de datos grandes en el Catálogo de datos para mejorar el rendimiento de las consultas. Implemente optimizaciones de rendimiento para los metadatos a los que se accede con frecuencia.
- Manténgase actualizado con la documentación y las prácticas recomendadas de AWS Glue: Consulte periódicamente la documentación y los recursos de AWS Glue para obtener las últimas novedades, prácticas recomendadas y recomendaciones. Asista a seminarios web, talleres y otros eventos de AWS Glue para aprender de los expertos y mantenerse informado sobre las nuevas características y capacidades.

AWS Glue Schema Registry

> **Note**
>
> El registro de esquemas de AWS Glue no se admite en la consola de AWS Glue en las siguientes regiones: Asia Pacífico (Yakarta) y Medio Oriente (EAU).

AWS Glue Schema Registry es una nueva característica que le permite descubrir, controlar y evolucionar de forma centralizada esquemas de flujo de datos. Un esquema define la estructura y el formato de un registro de datos. Con AWS Glue Schema Registry, puede administrar y aplicar esquemas en sus aplicaciones de streaming de datos mediante integraciones convenientes con Apache Kafka, Amazon Managed Streaming for Apache Kafka, Amazon Kinesis Data Streams, Amazon Managed Service para Apache Flink y AWS Lambda.

El registro de esquemas AWS Glue admite el formato de datos AVRO (v1.10.2), el formato de datos JSON con formato de esquemas JSON para el esquema (especificaciones Draft-04, Draft-06 y Draft-07) con la validación de esquema JSON mediante el método biblioteca de Everit, y admite los búferes de protocolo (Protobuf) en las versiones proto2 y proto3 sin soporte para `extensions` o `groups`, y el lenguaje Java, con otros formatos de datos e idiomas que estarán disponibles en el futuro. Las características soportadas incluyen compatibilidad, fuentes de esquemas mediante metadatos, registro automático de esquemas, compatibilidad con IAM y compresión ZLIB opcional para reducir el almacenamiento y la transferencia de datos. AWS Glue Schema Registry no precisa servidor y es gratuito.

El uso de un esquema como contrato de formato de datos entre productores y consumidores conduce a una mejor gobernanza de los datos, datos de mayor calidad y permite que los consumidores de datos sean resistentes a los cambios posteriores compatibles.

Schema Registry permite que sistemas dispares compartan un esquema para la serialización y la deserialización. Por ejemplo, suponga que tiene un productor y un consumidor de datos. El productor conoce el esquema cuando publica los datos. Schema Registry proporciona un serializador y deserializador para determinados sistemas como Amazon MSK o Apache Kafka.

Para obtener más información, consulte Cómo funciona Schema Registry.

Temas

- Schemas

- Registries
- Compatibilidad y control de versiones de esquemas
- Bibliotecas Serde de código abierto
- Cuotas del Schema Registry
- Cómo funciona Schema Registry
- Introducción a Schema Registry
- Integración con AWS Glue Schema Registry
- Migración desde un registro de esquemas de terceros a AWS Glue Schema Registry

Schemas

Un esquema define la estructura y el formato de un registro de datos. Un esquema es una especificación versionada para publicación, consumo o almacenamiento de confianza de datos.

En este esquema de ejemplo para Avro, el formato y la estructura están definidos por el diseño y los nombres de campos, y el formato de los nombres de campos está definido por los tipos de datos (por ejemplo, `string`, `int`).

```
{
    "type": "record",
    "namespace": "ABC_Organization",
    "name": "Employee",
    "fields": [
        {
            "name": "Name",
            "type": "string"
        },
        {
            "name": "Age",
            "type": "int"
        },
        {
            "name": "address",
            "type": {
                "type": "record",
                "name": "addressRecord",
                "fields": [
                    {
                        "name": "street",
```

```
                        "type": "string"
                    },
                    {
                        "name": "zipcode",
                        "type": "int"
                    }
                ]
            }
        }
    ]
}
```

En este ejemplo de esquema JSON Draft-07 para JSON, el formato está definido por la organización del esquema JSON.

```
{
 "$id": "https://example.com/person.schema.json",
 "$schema": "http://json-schema.org/draft-07/schema#",
 "title": "Person",
 "type": "object",
 "properties": {
  "firstName": {
   "type": "string",
   "description": "The person's first name."
  },
  "lastName": {
   "type": "string",
   "description": "The person's last name."
  },
  "age": {
   "description": "Age in years which must be equal to or greater than zero.",
   "type": "integer",
   "minimum": 0
  }
 }
}
```

En este ejemplo de Protobuf, el formato se define mediante laversión 2 del lenguaje de búferes de protocolo (proto2).

```
syntax = "proto2";
```

```
package tutorial;

option java_multiple_files = true;
option java_package = "com.example.tutorial.protos";
option java_outer_classname = "AddressBookProtos";

message Person {
  optional string name = 1;
  optional int32 id = 2;
  optional string email = 3;

  enum PhoneType {
    MOBILE = 0;
    HOME = 1;
    WORK = 2;
  }

  message PhoneNumber {
    optional string number = 1;
    optional PhoneType type = 2 [default = HOME];
  }

  repeated PhoneNumber phones = 4;
}

message AddressBook {
  repeated Person people = 1;
}
```

Registries

Un registro es un contenedor lógico de esquemas. Los registros le permiten organizar sus esquemas, así como administrar el control de acceso para sus aplicaciones. Un registro tiene un nombre de recurso de Amazon (ARN) que le permite organizar y establecer diferentes permisos de acceso a las operaciones de esquema dentro del registro.

Puede utilizar el registro predeterminado o crear tantos registros nuevos como sea necesario.

Jerarquía de AWS Glue Schema Registry

- RegistryName: [cadena]
  - RegistrYarn: [AWS ARN]

- CreatedTime: [marca de tiempo]
- UpdatedTime: [marca de tiempo]

- SchemaName: [cadena]
  - SchemaArn: [AWS ARN]
  - DataFormat: [Avro, Json o Protobuf]
  - Compatibilidad: [por ejemplo, BACKWARD (HACIA ATRÁS), BACKWARD_ALL (HACIA ATRÁS_TODO), FORWARD (HACIA ADELANTE), FORWARD_ALL (HACIA ADELANTE_TODO), FULL (COMPLETO), FULL_ALL (COMPLETO_TODO), NONE (NINGUNO), DISABLED (DESACTIVADO)]
  - Status: [por ejemplo, PENDING (PENDIENTE), AVAILABLE (DISPONIBLE), DELETING (ELIMINACIÓN)]
  - SchemaCheckPoint: [entero]
  - CreatedTime: [marca de tiempo]
  - UpdatedTime: [marca de tiempo]
- SchemaVersion: [cadena]
  - SchemaVersionNumber: [entero]
  - Status: [por ejemplo, PENDIENTE, DISPONIBLE, ELIMINACIÓN, ERROR]
  - SchemaDefinition: [cadena, valor: JSON]
  - CreatedTime: [marca de tiempo]
- SchemaVersionMetadata: [lista]
  - MetadataKey: [cadena]
  - MetadataInfo
  - MetadaValue: [cadena]
  - CreatedTime: [marca de tiempo]

Compatibilidad y control de versiones de esquemas

Cada esquema puede tener varias versiones. El control de versiones se rige por una regla de compatibilidad que se aplica a un esquema. Schema Registry contrasta las solicitudes para registrar nuevas versiones de esquema con esta regla antes de que puedan tener éxito.

Una versión de esquema marcada como punto de verificación se utiliza para determinar la compatibilidad de registrar nuevas versiones de un esquema. Cuando se crea un esquema por primera vez, el punto de verificación predeterminado será la primera versión. A medida que el esquema evoluciona con más versiones, puede utilizar CLI/SDK para cambiar el punto de control a una versión de un esquema mediante la herramienta API `UpdateSchema` que cumple con un conjunto de restricciones. En la consola, la edición de la definición de esquema o el modo de compatibilidad cambiará el punto de control a la última versión de forma predeterminada.

Los modos de compatibilidad permiten controlar cómo los esquemas pueden o no evolucionar con el tiempo. Estos modos forman el contrato entre las aplicaciones que producen y consumen datos. Cuando se envía una nueva versión de un esquema al registro, la regla de compatibilidad aplicada al nombre del esquema se utiliza para determinar si se puede aceptar la nueva versión. Existen ocho modos de compatibilidad: NONE (NINGUNO), DISABLED (DESACTIVADO), BACKWARD (HACIA ATRÁS), BACKWARD_ALL (HACIA ATRÁS_TODO), FORWARD (HACIA ADELANTE), FORWARD_ALL (HACIA ADELANTE_TODO), FULL (COMPLETO), FULL_ALL (COMPLETO_TODO).

En el formato de datos Avro, los campos pueden ser opcionales u obligatorios. Un campo opcional es aquel en el que el `Type` incluye nulo. Los campos obligatorios no tienen nulo como `Type`.

En el formato de datos Protobuf, los campos pueden ser opcionales (incluso repetidos) u obligatorios en la sintaxis proto2, mientras que todos los campos son opcionales (incluidos los repetidos) en la sintaxis proto3. Todas las reglas de compatibilidad se determinan en función de la comprensión de las especificaciones de los búferes de protocolo, así como en función de las pautas en la documentación sobre búferes de protocolo de Google.

- NONE (NINGUNO): no se aplica ningún modo de compatibilidad. Puede utilizar esta opción en escenarios de desarrollo o si no conoce los modos de compatibilidad que desea aplicar a los esquemas. Cualquier nueva versión que se agregue será aceptada sin someterse a una comprobación de compatibilidad.
- DISABLED (DESACTIVADO): esta opción de compatibilidad impide el control de versiones de un esquema en particular. No se pueden agregar nuevas versiones.
- BACKWARD (HACIA ATRÁS): se recomienda esta opción de compatibilidad ya que permite a los consumidores leer tanto la versión actual como la anterior del esquema. Puede utilizar esta opción para comprobar la compatibilidad con la última versión del esquema, al eliminar campos o agregar campos adicionales. Un caso de uso típico para HACIA ATRÁS es cuando la aplicación se ha creado para el esquema más reciente.

AVRO

Por ejemplo, suponga que tiene un esquema definido por nombre (obligatorio), apellido (obligatorio), email (obligatorio) y número de teléfono (opcional).

Si la siguiente versión del esquema elimina el campo de email obligatorio, esto se registraría correctamente. La compatibilidad HACIA ATRÁS requiere que los consumidores puedan leer la versión actual y anterior del esquema. Sus consumidores podrán leer el nuevo esquema a medida que se ignora el campo de email adicional de los mensajes antiguos.

Si tiene una nueva versión de esquema propuesta que agrega un campo obligatorio, por ejemplo, código postal, esto no se registrará correctamente con la compatibilidad HACIA ATRÁS. Los consumidores de la nueva versión no podrían leer mensajes antiguos antes de cambiar el esquema, ya que les faltará el campo de código postal obligatorio. Sin embargo, si el campo de código postal se estableció como opcional en el nuevo esquema, entonces la versión propuesta se registraría correctamente, ya que los consumidores pueden leer el esquema anterior sin el campo de código postal opcional.

JSON

Por ejemplo, supongamos que tiene una versión de esquema definida por nombre (opcional), apellido (opcional), email (opcional) y número de teléfono (opcional).

Si la siguiente versión de esquema agrega la propiedad opcional de número de teléfono, esto se registraría correctamente siempre y cuando la versión original del esquema no permita ninguna propiedad adicional al configurar el campo `additionalProperties` en falso. La compatibilidad HACIA ATRÁS requiere que los consumidores puedan leer la versión actual y anterior del esquema. Sus consumidores podrán leer los datos producidos con el esquema original en el que la propiedad del número de teléfono no existe.

Si tiene una nueva versión de esquema propuesta que agrega la propiedad de número de teléfono opcional, esto no se registrará correctamente con la compatibilidad HACIA ATRÁS, cuando la versión del esquema original establezca el campo `additionalProperties` a verdadero, es decir, cuando permita cualquier propiedad adicional. Los consumidores de la nueva versión no podrían leer mensajes antiguos antes de cambiar el esquema, ya que no pueden leer datos con propiedad de número de teléfono en un tipo diferente, por ejemplo cadena en lugar de número.

PROTOBUF

Por ejemplo, suponga que tiene una versión de esquema definida por un mensaje `Person` con `first name` (obligatorio), `last name` (obligatorio), `email` (obligatorio) y `phone number` (opcional) en la sintaxis proto2.

En forma semejante a las situaciones de AVRO, si la siguiente versión del esquema elimina el campo de `email` obligatorio, esto se registraría correctamente. La compatibilidad HACIA ATRÁS requiere que los consumidores puedan leer la versión actual y anterior del esquema. Sus consumidores podrán leer el nuevo esquema a medida que se ignora el campo de `email` adicional de los mensajes antiguos.

Si tiene una nueva versión de esquema propuesta que agrega un campo obligatorio, por ejemplo, `zip code`, esto no se registrará correctamente con la compatibilidad CON VERSIONES ANTERIORES. Los consumidores de la nueva versión no podrían leer mensajes antiguos antes de cambiar el esquema, ya que les faltará el campo de `zip code` obligatorio. Sin embargo, si el campo de `zip code` se estableció como opcional en el nuevo esquema, entonces la versión propuesta se registraría correctamente, ya que los consumidores pueden leer el esquema anterior sin el campo de `zip code` opcional.

En caso de un caso de uso de gRPC, agregar un nuevo servicio RPC o un método RPC es un cambio compatible con versiones anteriores. Por ejemplo, suponga que tiene una versión de esquema definida por un servicio RPC `MyService` con dos métodos RPC, `Foo` y `Bar`.

Si la próxima versión de esquema añade un nuevo método RPC llamado `Baz`, esto se registraría correctamente. Los consumidores podrán leer los datos producidos con el esquema original según la compatibilidad CON VERSIONES ANTERIORES desde el método RPC `Baz` recientemente agregado es opcional.

Si tiene una nueva versión de esquema propuesta que agrega un método RPC existente `Foo`, esto no se registrará correctamente con la compatibilidad CON VERSIONES ANTERIORES. Los consumidores de la nueva versión no podrían leer mensajes antiguos antes del cambio de esquema, ya que no pueden entender ni leer datos con el método RPC inexistente `Foo` en una aplicación de gRPC.

- BACKWARD_ALL (HACIA ATRÁS_TODO): esta opción de compatibilidad permite a los consumidores leer tanto la versión actual como todas las versiones anteriores del esquema. Puede utilizar esta opción para comprobar la compatibilidad con la última versión del esquema cuando elimine campos o agregue campos adicionales.

- FORWARD (HACIA ADELANTE): esta opción de compatibilidad permite a los consumidores leer tanto la versión actual como la inmediatamente siguiente del esquema, pero no necesariamente versiones posteriores. Puede utilizar esta opción para comprobar la compatibilidad con la última versión del esquema cuando elimine campos o agregue campos adicionales. Un caso de uso típico de HACIA ADELANTE es cuando la aplicación se ha creado para un esquema anterior y debe poder procesar un esquema más reciente.

  AVRO

  Por ejemplo, supongamos que tiene una versión de esquema definida por nombre (obligatorio), apellido (obligatorio), email (opcional).

  Si tiene una nueva versión de esquema que agrega un campo obligatorio, p. ej., número de teléfono, esto se registraría correctamente. La compatibilidad HACIA ADELANTE requiere que los consumidores puedan leer los datos producidos con el nuevo esquema utilizando la versión anterior.

  Si tiene una sugerencia de una versión de esquema que elimina el campo obligatorio de nombre, esto no se registrará correctamente con la compatibilidad HACIA ADELANTE. Los consumidores de la versión anterior no podrían leer los esquemas propuestos, ya que les falta el campo obligatorio de nombre. Sin embargo, si el campo de nombre original era opcional, entonces el nuevo esquema propuesto se registraría correctamente, ya que los consumidores pueden leer datos basados en el nuevo esquema que no incluye el campo opcional de nombre.

  JSON

  Por ejemplo, supongamos que tiene una versión de esquema definida por nombre (opcional), apellido (opcional), email (opcional) y número de teléfono (opcional).

  Si tiene una nueva versión de esquema que elimina la propiedad opcional de número de teléfono, esto se registraría correctamente siempre y cuando la nueva versión del esquema no permita ninguna propiedad adicional al configurar el campo `additionalProperties` en falso. La compatibilidad HACIA ADELANTE requiere que los consumidores puedan leer los datos producidos con el nuevo esquema utilizando la versión anterior.

  Si tiene una sugerencia de una versión de esquema que elimina la propiedad opcional de número de teléfono, esto no se registraría correctamente con la compatibilidad HACIA ADELANTE cuando la nueva versión del esquema establece el campo `additionalProperties` a verdadero, es decir, permite cualquier propiedad adicional. Los consumidores de la versión anterior no podrían

leer los esquemas propuestos, ya que podrían tener la propiedad de número de teléfono en un tipo diferente, por ejemplo cadena en lugar de número.

PROTOBUF

Por ejemplo, suponga que tiene una versión de esquema definida por un mensaje `Person` con `first name` (obligatorio), `last name`(obligatorio) y `email` (opcional) en la sintaxis proto2.

De manera semejante a las situaciones de AVRO, si tiene una nueva versión de esquema que agrega un campo obligatorio, p. ej., `phone number`, esto se registraría correctamente. La compatibilidad HACIA ADELANTE requiere que los consumidores puedan leer los datos producidos con el nuevo esquema utilizando la versión anterior.

Si tiene una sugerencia de una versión de esquema que elimina el campo obligatorio de `first name`, esto no se registrará correctamente con la compatibilidad CON VERSIONES FUTURAS. Los consumidores de la versión anterior no podrían leer los esquemas propuestos, ya que les falta el campo obligatorio de `first name`. Sin embargo, si el campo de `first name` original era opcional, entonces el nuevo esquema propuesto se registraría correctamente, ya que los consumidores pueden leer datos basados en el nuevo esquema que no incluye el campo opcional de `first name`.

En caso de un caso de uso de gRPC, quitar un servicio RPC o un método RPC es un cambio compatible con versiones futuras. Por ejemplo, suponga que tiene una versión de esquema definida por un servicio RPC `MyService` con dos métodos RPC, `Foo` y `Bar`.

Si la próxima versión del esquema elimina el método RPC existente denominado `Foo`, esto se registraría correctamente según la compatibilidad CON VERSIONES FUTURAS, ya que los consumidores pueden leer los datos producidos con el nuevo esquema utilizando la versión anterior. Si tiene una sugerencia de una nueva versión de esquema que agrega un método RPC de `Baz`, esto no se registrará correctamente con la compatibilidad CON VERSIONES FUTURAS. Los consumidores de la versión anterior no podrían leer los esquemas propuestos, ya que les falta el método RPC de `Baz`.

- FORWARD_ALL (HACIA ADELANTE_TODO): esta opción de compatibilidad permite a los consumidores leer datos escritos por los productores de cualquier nuevo esquema registrado. Puede utilizar esta opción cuando necesite agregar campos o eliminar campos opcionales, y comprobar la compatibilidad con todas las versiones de esquema anteriores.
- FULL (COMPLETO): esta opción de compatibilidad permite a los consumidores leer los datos escritos por los productores con la versión inmediatamente anterior o siguiente del esquema, pero

no necesariamente versiones anteriores o posteriores. Puede utilizar esta opción para comprobar la compatibilidad con la última versión del esquema cuando elimine campos o agregue campos adicionales.

- FULL_ALL (COMPLETO_TODO): esta opción de compatibilidad permite a los consumidores leer los datos escritos por los productores con todas las versiones de esquema anteriores. Puede utilizar esta opción para comprobar la compatibilidad con todas las versiones anteriores del esquema, cuando agregue o elimine campos opcionales.

Bibliotecas Serde de código abierto

AWS proporciona bibliotecas Serde de código abierto como marco para serializar y deserializar datos. El diseño de código abierto de estas bibliotecas permite que las aplicaciones y marcos de código abierto comunes soporten estas bibliotecas en sus proyectos.

Para obtener más detalles sobre cómo funcionan las bibliotecas de Serde, consulte Cómo funciona Schema Registry.

Cuotas del Schema Registry

Las cuotas, también conocidas como límites en AWS, son el valor máximo de los recursos, acciones y elementos de su cuenta de AWS. Los siguientes son los límites flexibles para el Schema Registry en AWS Glue.

Pares de clave-valor de metadatos de la versión del esquema

Puede tener hasta 10 pares clave-valor por SchemaVersion por región de AWS.

Puede ver o establecer los pares de metadatos clave-valor con las API QuerySchemaVersionMetadata acción (Python: query_schema_version_metadata) o PutSchemaVersionMetadata acción (Python: put_schema_version_metadata).

Los siguientes son los límites invariables para el Schema Registry en AWS Glue.

Registries

Puede tener hasta 100 registros por región de AWS para esta cuenta.

SchemaVersion

Puede tener hasta 10 000 versiones de esquema por región de AWS para esta cuenta.

Cada esquema nuevo crea una nueva versión de esquema, por lo que teóricamente puede tener hasta 10 000 esquemas por cuenta por región, si es que cada esquema tiene una sola versión.

Cargas de esquema

Hay un límite de tamaño de 170 KB para las cargas de esquema.

Cómo funciona Schema Registry

En esta sección se describe cómo funcionan los procesos de serialización y deserialización en Schema Registry.

1. Registrar un esquema: si el esquema aún no existe en el registro, puede registrarse con un nombre de esquema igual al nombre del destino (por ejemplo, test_topic, test_stream, prod_firehose) o el productor puede proporcionar un nombre personalizado para el esquema. Los productores también pueden agregar pares clave-valor al esquema como metadatos, como orígenes: msk_kafka_topic_a, o aplicar etiquetas AWS a esquemas en la creación de esquemas. Una vez registrado un esquema, el Schema Registry devuelve el ID de versión del esquema al serializador. Si el esquema existe pero el serializador está utilizando una nueva versión que no existe, Schema Registry comprobará la referencia del esquema en función de una regla de compatibilidad para asegurarse de que la nueva versión es compatible antes de registrarla como una nueva versión.

   Existen dos métodos para registrar un esquema: registro manual y registro automático. Puede registrar un esquema manualmente a través de la consola de AWS Glue o CLI/SDK.

   Cuando el registro automático está activado en la configuración del serializador, se realizará el registro automático del esquema. Si no se proporciona el `REGISTRY_NAME` en las configuraciones del productor, entonces el registro automático registrará la nueva versión del esquema según el registro predeterminado (default-registry). Consulte Instalación de bibliotecas SerDe para obtener información sobre cómo especificar la propiedad de registro automático.
2. El serializador valida los registros de datos respecto del esquema: cuando la aplicación que produce datos ha registrado su esquema, el serializador de Schema Registry valida que el registro que está produciendo la aplicación se estructure con los campos y tipos de datos que coincidan con un esquema registrado. Si el esquema del registro no coincide con un esquema registrado, el serializador devolverá una excepción y la aplicación no entregará el registro al destino.

Si no existe ningún esquema y si el nombre del esquema no se proporciona a través de las configuraciones del productor, el esquema se crea con el mismo nombre que el nombre del tema (si es Apache Kafka o Amazon MSK) o el nombre del flujo (si es Kinesis Data Streams).

Cada registro tiene una definición de esquema y datos. La definición de esquema se consulta respecto de los esquemas y versiones existentes en el Schema Registry.

De forma predeterminada, los productores almacenan en caché las definiciones de esquema y los ID de versión de esquema de los esquemas registrados. Si la definición de versión de esquema de un registro no coincide con lo que está disponible en caché, el productor intentará validar el esquema con Schema Registry. Si la versión del esquema es válida, su ID de versión y definición se almacenarán en caché de manera local en el productor.

Puede ajustar el período de caché predeterminado (24 horas) dentro de las propiedades del productor opcionales en el paso n.º 3 de Instalación de bibliotecas SerDe.

3. Serializar y entregar registros: si el registro cumple con el esquema, el serializador agrega a cada registro el ID de la versión del esquema, serializa el registro en función del formato de datos seleccionado (AVRO, JSON, Protobuf u otros formatos próximamente), comprime el registro (configuración opcional del productor) y lo entrega al destino.
4. Los consumidores deserializan los datos: los consumidores que leen estos datos utilizan la biblioteca del deserializador de Schema Registry que analiza el ID de versión del esquema proveniente de la carga del registro.
5. El deserializador puede solicitar el esquema desde el Schema Registry: si es la primera vez que el deserializador ha visto registros con un identificador de versión de esquema concreto, a través del ID de versión de esquema, el deserializador solicitará el esquema desde Schema Registry y almacenará el esquema localmente en el consumidor. Si Schema Registry no puede deserializar el registro, el consumidor puede registrar los datos a partir del registro y continuar, o detener la aplicación.
6. El deserializador utiliza el esquema para deserializar el registro: cuando el deserializador recupera el ID de versión del esquema de Schema Registry, descomprime el registro (si el registro enviado por el productor está comprimido) y utiliza el esquema para deserializar el registro. La aplicación procesa ahora el registro.

> **Note**
>
> Cifrado: sus clientes se comunican con Schema Registry a través de llamadas a la API que cifran los datos en tránsito mediante el cifrado TLS a través de HTTPS. Los esquemas almacenados en Schema Registry siempre se cifran en reposo mediante una clave AWS Key Management Service (AWS KMS) administrada por el servicio.

> **Note**
>
> Autorización de usuario: el registro de esquemas admite políticas de IAM basadas en identidad.

Introducción a Schema Registry

Las siguientes secciones ofrecen información general y una guía para configurar y empezar a utilizar Schema Registry. Para obtener más información sobre los conceptos y componentes de Schema Registry, consulte AWS Glue Schema Registry.

Temas

- Instalación de bibliotecas SerDe
- Uso de AWS CLI para las API de AWS Glue Schema Registry
- Creación de un registro
- Trabajar con un registro específico (JAVA POJO) para JSON
- Creación de un esquema
- Actualización de un esquema o registro
- Eliminación de un esquema o registro
- Ejemplos de IAM para serializadores
- Ejemplos de IAM para deserializadores
- Conectividad privada mediante AWS PrivateLink
- Acceso a métricas de Amazon CloudWatch
- Ejemplo de plantilla de AWS CloudFormation para Schema Registry

Instalación de bibliotecas SerDe

> **Note**
>
> Requisitos previos: antes de completar los siguientes pasos, deberá tener un clúster de Amazon Managed Streaming for Apache Kafka (Amazon MSK) o Apache Kafka en ejecución. Sus productores y consumidores deben ejecutarse en Java 8 o superior.

Las bibliotecas SerDe proporcionan un marco para serializar y deserializar datos.

Instalará el serializador de código abierto para sus aplicaciones que producen datos (colectivamente los “serializadores”). El serializador controla la serialización, la compresión y la interacción con Schema Registry. El serializador extrae automáticamente el esquema de un registro que se está escribiendo en un destino compatible con Schema Registry, como Amazon MSK. Del mismo modo, instalará el deserializador de código abierto en sus aplicaciones que consumen datos.

Para instalar las bibliotecas en productores y consumidores:

1. Dentro de los archivos pom.xml de los productores y consumidores, agregue esta dependencia a través del siguiente código:

   ```
   <dependency>
       <groupId>software.amazon.glue</groupId>
       <artifactId>schema-registry-serde</artifactId>
       <version>1.1.5</version>
   </dependency>
   ```

   También puede clonar el repositorio Github de AWS Glue Schema Registry.

2. Configure sus productores con estas propiedades requeridas:

   ```
   props.put(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG,
    StringSerializer.class.getName()); // Can replace StringSerializer.class.getName())
    with any other key serializer that you may use
   props.put(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG,
    GlueSchemaRegistryKafkaSerializer.class.getName());
   props.put(AWSSchemaRegistryConstants.AWS_REGION, "us-east-2");
   properties.put(AWSSchemaRegistryConstants.DATA_FORMAT, "JSON"); // OR "AVRO"
   ```

Si no hay esquemas existentes, debe activarse el registro automático (siguiente paso). Si tiene un esquema que desea aplicar, reemplace “my-schema (mi esquema)” con su nombre de esquema. También debe proporcionarse el “registry-name (nombre del registro)” si el registro automático del esquema está desactivado. Si el esquema se crea como “default-registry (registro predeterminado)”, entonces el nombre del registro se puede omitir.

3. (Opcional) configure cualquiera de estas propiedades de productor opcionales. Para ver descripciones detalladas de propiedades, consulte el archivo ReadMe (Léame).

```
props.put(AWSSchemaRegistryConstants.SCHEMA_AUTO_REGISTRATION_SETTING, "true"); // If
 not passed, uses "false"
props.put(AWSSchemaRegistryConstants.SCHEMA_NAME, "my-schema"); // If not passed,
 uses transport name (topic name in case of Kafka, or stream name in case of Kinesis
 Data Streams)
props.put(AWSSchemaRegistryConstants.REGISTRY_NAME, "my-registry"); // If not passed,
 uses "default-registry"
props.put(AWSSchemaRegistryConstants.CACHE_TIME_TO_LIVE_MILLIS, "86400000"); // If
 not passed, uses 86400000 (24 Hours)
props.put(AWSSchemaRegistryConstants.CACHE_SIZE, "10"); // default value is 200
props.put(AWSSchemaRegistryConstants.COMPATIBILITY_SETTING, Compatibility.FULL); //
 Pass a compatibility mode. If not passed, uses Compatibility.BACKWARD
props.put(AWSSchemaRegistryConstants.DESCRIPTION, "This registry is used for several
 purposes."); // If not passed, constructs a description
props.put(AWSSchemaRegistryConstants.COMPRESSION_TYPE,
 AWSSchemaRegistryConstants.COMPRESSION.ZLIB); // If not passed, records are sent
 uncompressed
```

El registro automático registra la versión del esquema en el registro predeterminado (“default-registry”). Si no se especifica un `SCHEMA_NAME` en el paso anterior, entonces el nombre del tema se infiere como el `SCHEMA_NAME`.

Consulte Compatibilidad y control de versiones de esquemas para obtener más información sobre los modos de compatibilidad.

4. Configure sus consumidores con estas propiedades obligatorias:

```
props.put(ConsumerConfig.KEY_DESERIALIZER_CLASS_CONFIG,
 StringDeserializer.class.getName());
props.put(ConsumerConfig.VALUE_DESERIALIZER_CLASS_CONFIG,
 GlueSchemaRegistryKafkaDeserializer.class.getName());
```

```
props.put(AWSSchemaRegistryConstants.AWS_REGION, "us-east-2"); // Pass an Región de
 AWS
props.put(AWSSchemaRegistryConstants.AVRO_RECORD_TYPE,
 AvroRecordType.GENERIC_RECORD.getName()); // Only required for AVRO data format
```

5. (Opcional) configure estas propiedades de consumidores opcionales. Para ver descripciones detalladas de propiedades, consulte el archivo ReadMe (Léame).

```
properties.put(AWSSchemaRegistryConstants.CACHE_TIME_TO_LIVE_MILLIS, "86400000"); //
 If not passed, uses 86400000
props.put(AWSSchemaRegistryConstants.CACHE_SIZE, "10"); // default value is 200
props.put(AWSSchemaRegistryConstants.SECONDARY_DESERIALIZER,
 "com.amazonaws.services.schemaregistry.deserializers.external.ThirdPartyDeserializer"); //
 For migration fall back scenario
```

Uso de AWS CLI para las API de AWS Glue Schema Registry

Para utilizar AWS CLI para las API de AWS Glue Schema Registry, asegúrese de actualizar su AWS CLI a la versión más reciente.

Creación de un registro

Puede utilizar el registro predeterminado o crear tantos registros nuevos como sea necesario mediante las API de AWS Glue o la consola de AWS Glue.

Las API de AWS Glue

Puede seguir estos pasos para realizar esta tarea mediante las API de AWS Glue.

Para agregar un registro nuevo, use la API de CreateRegistry acción (Python: create_registry). Especifique `RegistryName` como el nombre del registro que se va a crear con una longitud máxima de 255, y solo puede contener letras, números, guión, guión bajo, signo de dólar o marca hash.

Especifique una `Description` como una cadena cuya extensión no sea más de 2048 bytes y que coincida con el patrón de cadena de varias líneas de la dirección URI.

De manera opcional, especifique una o varias `Tags` para su registro, como matriz de mapas de pares clave-valor.

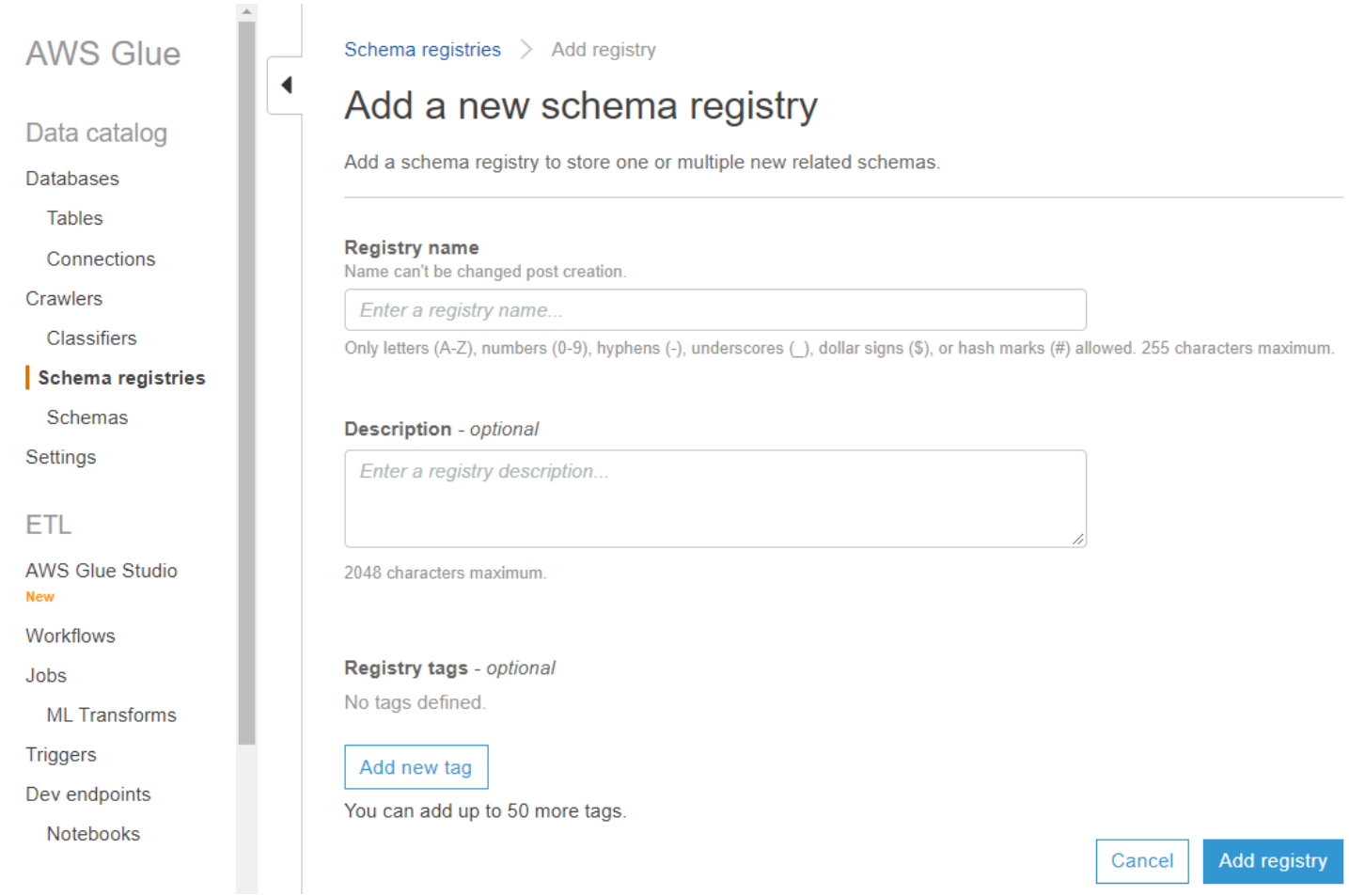
```
aws glue create-registry --registry-name registryName1 --description description
```

Cuando se crea el registro se le asigna un nombre de recurso de Amazon (ARN). Este ARN lo puede ver en `RegistryArn` de la respuesta de la API. Ahora que ha creado un registro, cree uno o más esquemas para ese registro.

Consola de AWS Glue

Para agregar un nuevo registro en la consola de AWS Glue:

1. Inicie sesión en AWS Management Console y abra la consola de AWS Glue en https://console.aws.amazon.com/glue/.
2. En el panel de navegación, en Data catalog (Catálogo de datos), elija Schema registries (Registros de esquemas).
3. Elija Add registry (Agregar registro).
4. Escriba un Registry name (Nombre del Registro) para el registro, que conste de letras, números, guiones o guiones bajos. Este nombre no se puede cambiar.
5. Escriba una Description (Descripción) (opcional) para el registro.
6. De manera opcional, aplique una o varias etiquetas a su registro. Elija Add new tag (Agregar nueva etiqueta) y especifique una Tag key (Clave de etiqueta) y, opcionalmente, un Tag value (Valor de etiqueta).
7. Elija Add registry (Agregar registro).



Cuando se crea el registro, se le asigna un nombre de recurso de Amazon (ARN), que puede ver al seleccionar el registro de la lista en Schema registries (Registros de esquemas). Ahora que ha creado un registro, cree uno o más esquemas para ese registro.

Trabajar con un registro específico (JAVA POJO) para JSON

Puede usar un objeto Java antiguo simple (POJO) y transferir el objeto como un registro. Esto es similar a la noción de un registro específico en AVRO. mbknor-jackson-jsonschema puede generar un esquema JSON para el POJO transferido. Esta biblioteca también puede inyectar información adicional en el esquema JSON.

La biblioteca de AWS Glue Schema Registry utiliza el campo “className (nombre de clase)” inyectado en el esquema para proporcionar un nombre de clase completamente clasificado. El deserializador utiliza el campo “className” para deserializar en un objeto de esa clase.

```
 Example class :

@JsonSchemaDescription("This is a car")
```

```
@JsonSchemaTitle("Simple Car Schema")
@Builder
@AllArgsConstructor
@EqualsAndHashCode
// Fully qualified class name to be added to an additionally injected property
// called className for deserializer to determine which class to deserialize
// the bytes into
@JsonSchemaInject(
        strings = {@JsonSchemaString(path = "className",
                value =
 "com.amazonaws.services.schemaregistry.integrationtests.generators.Car")}
)
// List of annotations to help infer JSON Schema are defined by https://github.com/
mbknor/mbknor-jackson-jsonSchema
public class Car {
    @JsonProperty(required = true)
    private String make;

    @JsonProperty(required = true)
    private String model;

    @JsonSchemaDefault("true")
    @JsonProperty
    public boolean used;

    @JsonSchemaInject(ints = {@JsonSchemaInt(path = "multipleOf", value = 1000)})
    @Max(200000)
    @JsonProperty
    private int miles;

    @Min(2000)
    @JsonProperty
    private int year;

    @JsonProperty
    private Date purchaseDate;

    @JsonProperty
    @JsonFormat(shape = JsonFormat.Shape.NUMBER)
    private Date listedDate;

    @JsonProperty
    private String[] owners;
```

```
    @JsonProperty
    private Collection<Float> serviceChecks;

    // Empty constructor is required by Jackson to deserialize bytes
    // into an Object of this class
    public Car() {}
}
```

Creación de un esquema

Puede crear un esquema con las API de AWS Glue o la consola de AWS Glue.

Las API de AWS Glue

Puede seguir estos pasos para realizar esta tarea mediante las API de AWS Glue.

Para agregar un esquema nuevo, use la API de CreateSchema acción (Python: create_schema).

Especifique una estructura de `RegistryId` para indicar un registro para el esquema. O bien, omita el `RegistryId` para utilizar el registro predeterminado.

Especifique un `SchemaName`, que conste de letras, números, guiones o guiones bajos, y `DataFormat` como **`AVRO`** o **`JSON`**. Una vez configurado el `DataFormat` en un esquema no puede modificarse.

Especifique un modo de `Compatibility`:

- Hacia atrás (recomendado): el consumidor puede leer tanto la versión actual como la anterior.
- Hacia atrás todo: el consumidor puede leer tanto la versión actual como todas las anteriores.
- Hacia adelante: el consumidor puede leer tanto la versión actual como la posterior.
- Hacia adelante todo: el consumidor puede leer tanto la versión actual como todas las versiones posteriores.
- Completo: combinación de hacia atrás y hacia adelante.
- Completo todo: combinación de hacia atrás todos y hacia adelante todos.
- Ninguno: no se realizan comprobaciones de compatibilidad.
- Desactivado: evita cualquier control de versiones para este esquema.

Opcionalmente, especifique `Tags` para su esquema.

Especifique una `SchemaDefinition` para definir el esquema en formato de datos Avro, JSON o Protobuf. Consulte estos ejemplos.

Para el formato de datos Avro:

```
aws glue create-schema --registry-id RegistryName="registryName1" --schema-name
 testschema --compatibility NONE --data-format AVRO --schema-definition "{\"type\":
 \"record\", \"name\": \"r1\", \"fields\": [ {\"name\": \"f1\", \"type\": \"int\"},
 {\"name\": \"f2\", \"type\": \"string\"} ]}"
```

```
aws glue create-schema --registry-id RegistryArn="arn:aws:glue:us-
east-2:901234567890:registry/registryName1" --schema-name testschema --compatibility
 NONE --data-format AVRO  --schema-definition "{\"type\": \"record\", \"name\": \"r1\",
 \"fields\": [ {\"name\": \"f1\", \"type\": \"int\"}, {\"name\": \"f2\", \"type\":
 \"string\"} ]}"
```

Para el formato de datos JSON:

```
aws glue create-schema --registry-id RegistryName="registryName" --schema-name
 testSchemaJson --compatibility NONE --data-format JSON --schema-definition "{\"$schema
\": \"http://json-schema.org/draft-07/schema#\",\"type\":\"object\",\"properties\":
{\"f1\":{\"type\":\"string\"}}}"
```

```
aws glue create-schema --registry-id RegistryArn="arn:aws:glue:us-
east-2:901234567890:registry/registryName" --schema-name testSchemaJson --compatibility
 NONE --data-format JSON --schema-definition "{\"$schema\": \"http://json-schema.org/
draft-07/schema#\",\"type\":\"object\",\"properties\":{\"f1\":{\"type\":\"string\"}}}"
```

Para el formato de datos Protobuf:

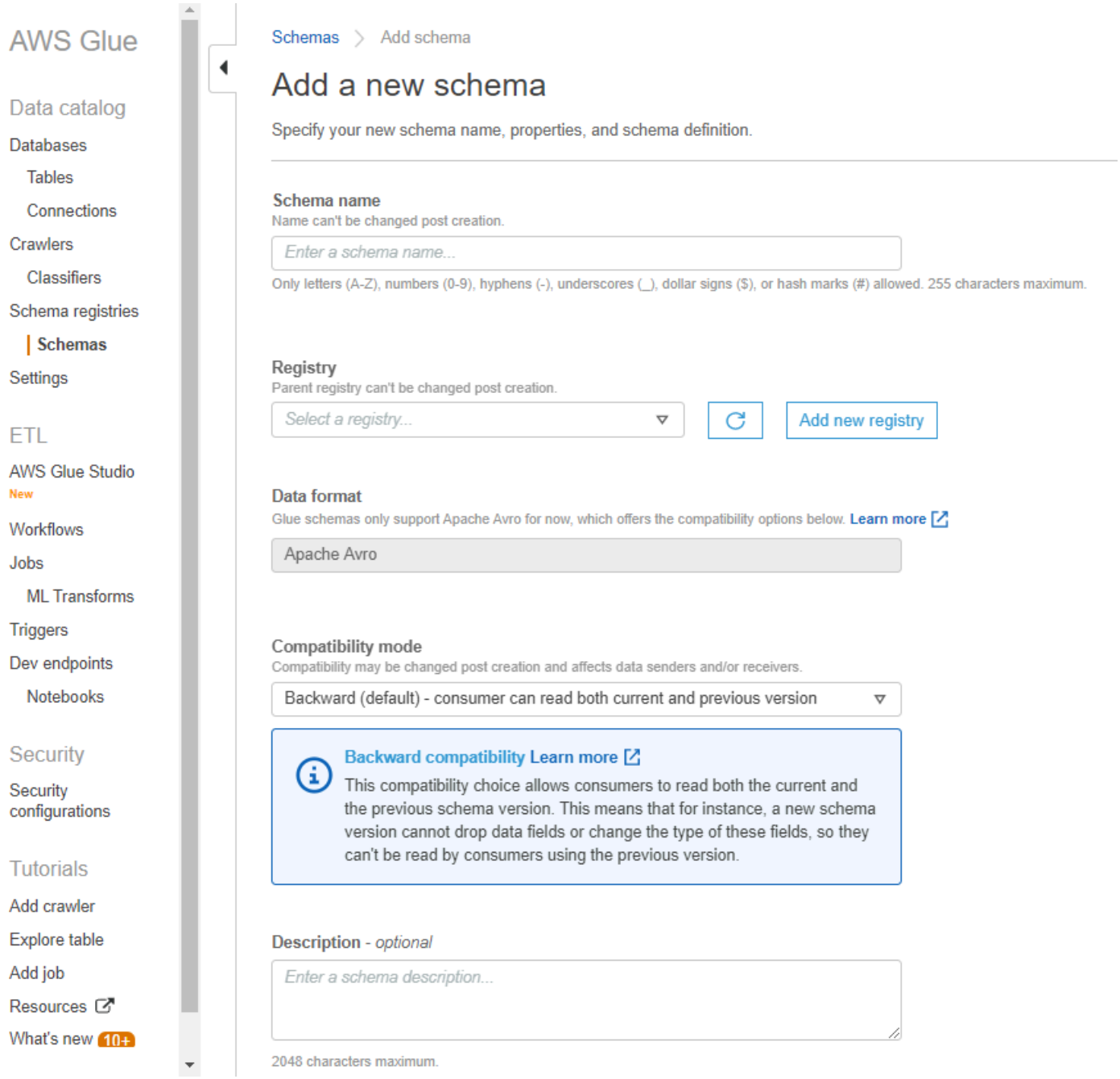
```
aws glue create-schema --registry-id RegistryName="registryName" --schema-name
 testSchemaProtobuf --compatibility NONE --data-format PROTOBUF --schema-definition
 "syntax = \"proto2\";package org.test;message Basic { optional int32 basic = 1;}"
```

```
aws glue create-schema --registry-id RegistryArn="arn:aws:glue:us-
east-2:901234567890:registry/registryName" --schema-name testSchemaProtobuf
 --compatibility NONE --data-format PROTOBUF --schema-definition "syntax =
 \"proto2\";package org.test;message Basic { optional int32 basic = 1;}"
```

Consola de AWS Glue

Para agregar un nuevo esquema a través de la consola de AWS Glue:

1. Inicie sesión en la consola de administración de AWS y abra la consola de AWS Glue en https://console.aws.amazon.com/glue/.
2. En el panel de navegación, en Data catalog (Catálogo de datos), elija Schema (Esquemas).
3. Elija Add schema (Agregar esquema).
4. Escriba un Schema name (Nombre del esquema), que conste de letras, números, guiones, guiones bajos, signos de dólar o marcas hash. Este nombre no se puede cambiar.
5. Elija el Registry (Registro) en el que se almacenará el esquema del menú desplegable. El registro principal no se puede cambiar después de la creación.
6. Deje el Data format (Formato de los datos) como Apache Avro o JSON. Este formato se aplica a todas las versiones de este esquema.
7. Elija un (Compatibility mode) Modo de compatibilidad.
   - Hacia atrás (recomendado): el receptor puede leer tanto la versión actual como la anterior.
   - Hacia atrás todo: el receptor puede leer tanto la versión actual como todas las anteriores.
   - Hacia adelante: el remitente puede escribir tanto la versión actual como la anterior.
   - Hacia adelante todo: el remitente puede escribir tanto la versión actual como todas las versiones anteriores.
   - Completo: combinación de hacia atrás y hacia adelante.
   - Completo todo: combinación de hacia atrás todos y hacia adelante todos.
   - Ninguno: no se realizan comprobaciones de compatibilidad.
   - Desactivado: evita cualquier control de versiones para este esquema.
8. Escriba una Description (Descripción) para el registro de hasta 250 caracteres.



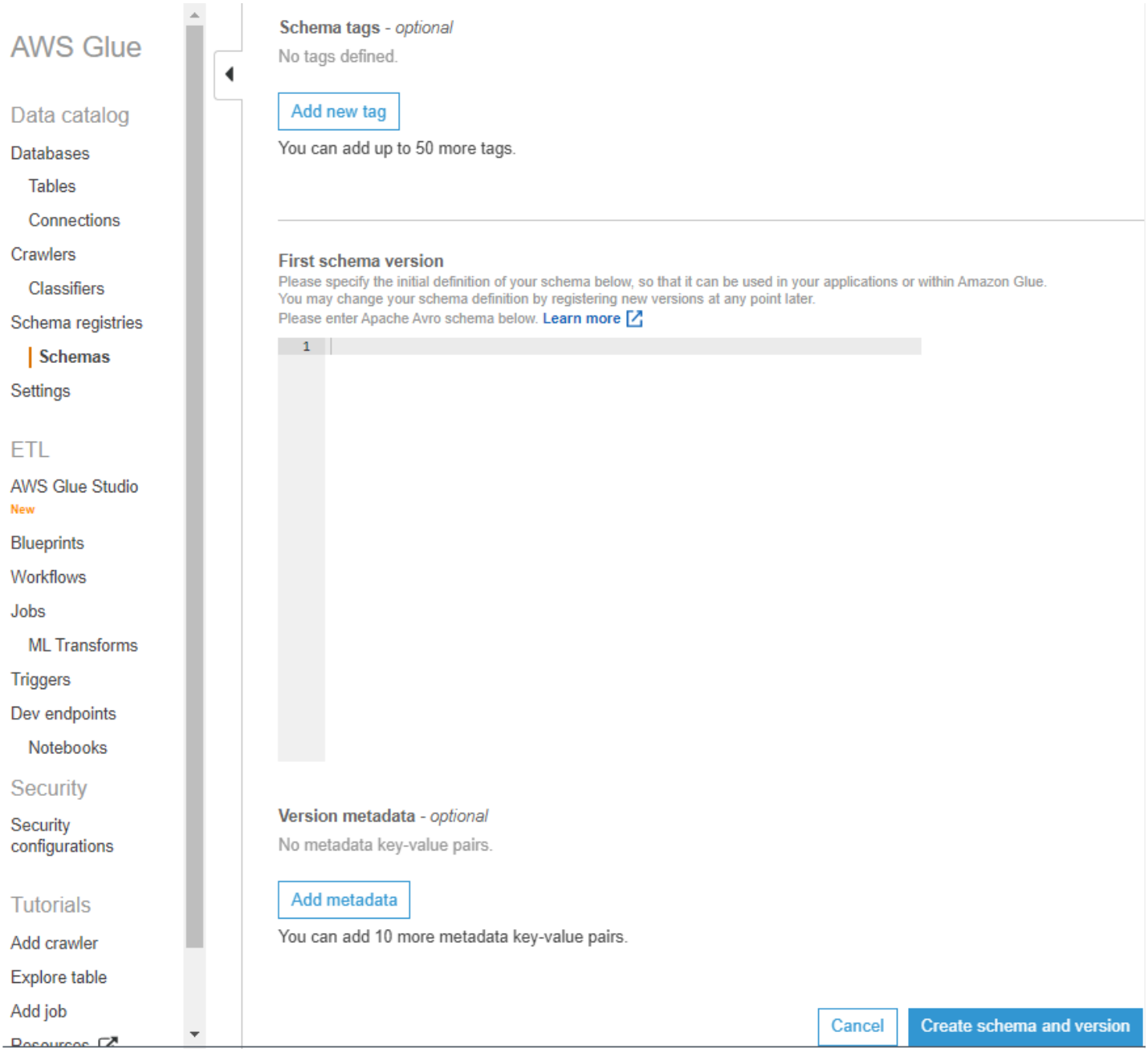
9. De manera opcional, aplique una o varias etiquetas a su esquema. Elija Add new tag (Agregar nueva etiqueta) y especifique una Tag key (Clave de etiqueta) y, opcionalmente, un Tag value (Valor de etiqueta).
10. En la casilla, First schema version (Primera versión del esquema), ingrese o pegue el esquema inicial.

    Para el formato Avro, consulte Trabajar con formato de datos Avro

Para el formato JSON, consulte Trabajar con formato de datos JSON

11. También puede seleccionar Add metadata (Agregar metadatos) para agregar metadatos de versión para anotar o clasificar la versión del esquema.

12. Elija Create schema and version (Crear esquema y versión).



El esquema se crea y aparece en la lista en Schemas (Esquemas).

Trabajar con formato de datos Avro

Avro proporciona servicios de serialización e intercambio de datos. Avro almacena la definición de datos en formato JSON, lo que facilita su lectura e interpretación. Los datos en sí se almacenan en formato binario.

Para obtener información sobre cómo definir un esquema Apache Avro, consulte la especificación de Apache Avro.

Trabajar con formato de datos JSON

Los datos se pueden serializar con formato JSON. Formato de esquemas JSON define el estándar para el formato de esquema JSON.

Actualización de un esquema o registro

Una vez creado, puede editar los esquemas, las versiones de esquema o el registro.

Actualización de un registro

Puede actualizar un registro mediante las API de AWS Glue o la consola de AWS Glue. No se puede editar el nombre de un registro existente. La descripción de un registro puede editarse.

Las API de AWS Glue

Para actualizar un registro existente, use la API UpdateRegistry acción (Python: update_registry).

Especifique una estructura de `RegistryId` para indicar el registro que desea actualizar. Transferir una `Description` para cambiar la descripción de un registro.

```
aws glue update-registry --description updatedDescription --registry-id
 RegistryArn="arn:aws:glue:us-east-2:901234567890:registry/registryName1"
```

Consola de AWS Glue

Para actualizar un registro mediante la consola de AWS Glue:

1. Inicie sesión en AWS Management Console y abra la consola de AWS Glue en https://console.aws.amazon.com/glue/.
2. En el panel de navegación, en Data catalog (Catálogo de datos), elija Schema registries (Registros de esquemas).

3. Seleccione un registro de la lista de registros, al marcar la casilla correspondiente.
4. En el menú Action (Acción), elija Edit registry (Editar registro).

Actualización de un esquema

Puede actualizar la descripción o la configuración de compatibilidad de un esquema.

Para actualizar un esquema existente, use la API UpdateSchema acción (Python: update_schema).

Especifique una estructura de `SchemaId` para indicar el esquema que desea actualizar. Uno de `VersionNumber` o `Compatibility` debe ser proporcionado.

Ejemplos de código 11:

```
aws glue update-schema --description testDescription --schema-id
 SchemaName="testSchema1",RegistryName="registryName1" --schema-version-number
 LatestVersion=true --compatibility NONE
```

```
aws glue update-schema --description testDescription --schema-id
 SchemaArn="arn:aws:glue:us-east-2:901234567890:schema/registryName1/testSchema1" --
schema-version-number LatestVersion=true --compatibility NONE
```

Agregar una versión de esquema.

Cuando agregue una versión de esquema, deberá comparar las versiones para asegurarse de que se aceptará el nuevo esquema.

Para agregar una nueva versión a un esquema existente, utilice la API RegisterSchemaVersion acción (Python: register_schema_version).

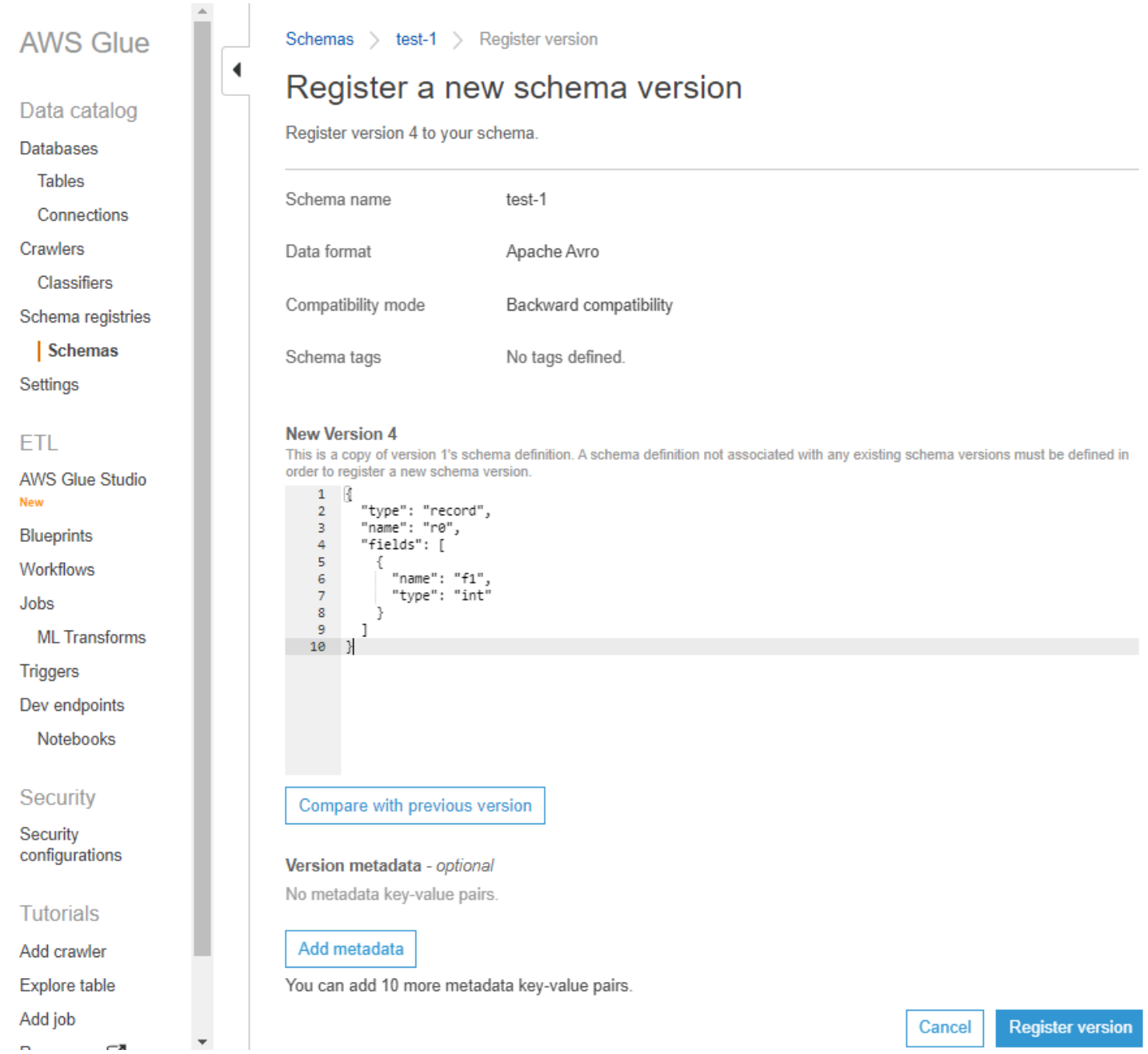
Especifique una estructura de `SchemaId` para indicar el esquema para el que desea agregar una versión, y una `SchemaDefinition` para definir el esquema.

Ejemplos de código 12:

```
aws glue register-schema-version --schema-definition "{\"type\": \"record\", \"name\":
 \"r1\", \"fields\": [ {\"name\": \"f1\", \"type\": \"int\"}, {\"name\": \"f2\", \"type
\": \"string\"} ]}" --schema-id SchemaArn="arn:aws:glue:us-east-1:901234567890:schema/
registryName/testschema"
```

```
aws glue register-schema-version --schema-definition "{\"type\": \"record\", \"name\":
 \"r1\", \"fields\": [ {\"name\": \"f1\", \"type\": \"int\"}, {\"name\": \"f2\", \"type
\": \"string\"} ]}" --schema-id SchemaName="testschema",RegistryName="testregistry"
```

1. Inicie sesión en AWS Management Console y abra la consola de AWS Glue en https://console.aws.amazon.com/glue/.
2. En el panel de navegación, en Data catalog (Catálogo de datos), elija Schema (Esquemas).
3. Seleccione el esquema de la lista de esquemas, al marcar la casilla correspondiente.
4. Seleccione uno o varios esquemas de la lista, al marcar las casillas correspondientes.
5. En el menú Action (Acción), elija Register new version (Registrar nueva versión).
6. En la casilla New version (Nueva versión), ingrese o pegue el nuevo esquema.
7. Elija Compare with previous version (Comparar con la versión anterior) para ver las diferencias con la versión del esquema anterior.
8. También puede seleccionar Add metadata (Agregar metadatos) para agregar metadatos de versión para anotar o clasificar la versión del esquema. Ingrese Key (Clave) y Value (Valor) opcional.
9. Elija Register version (Registrar versión).



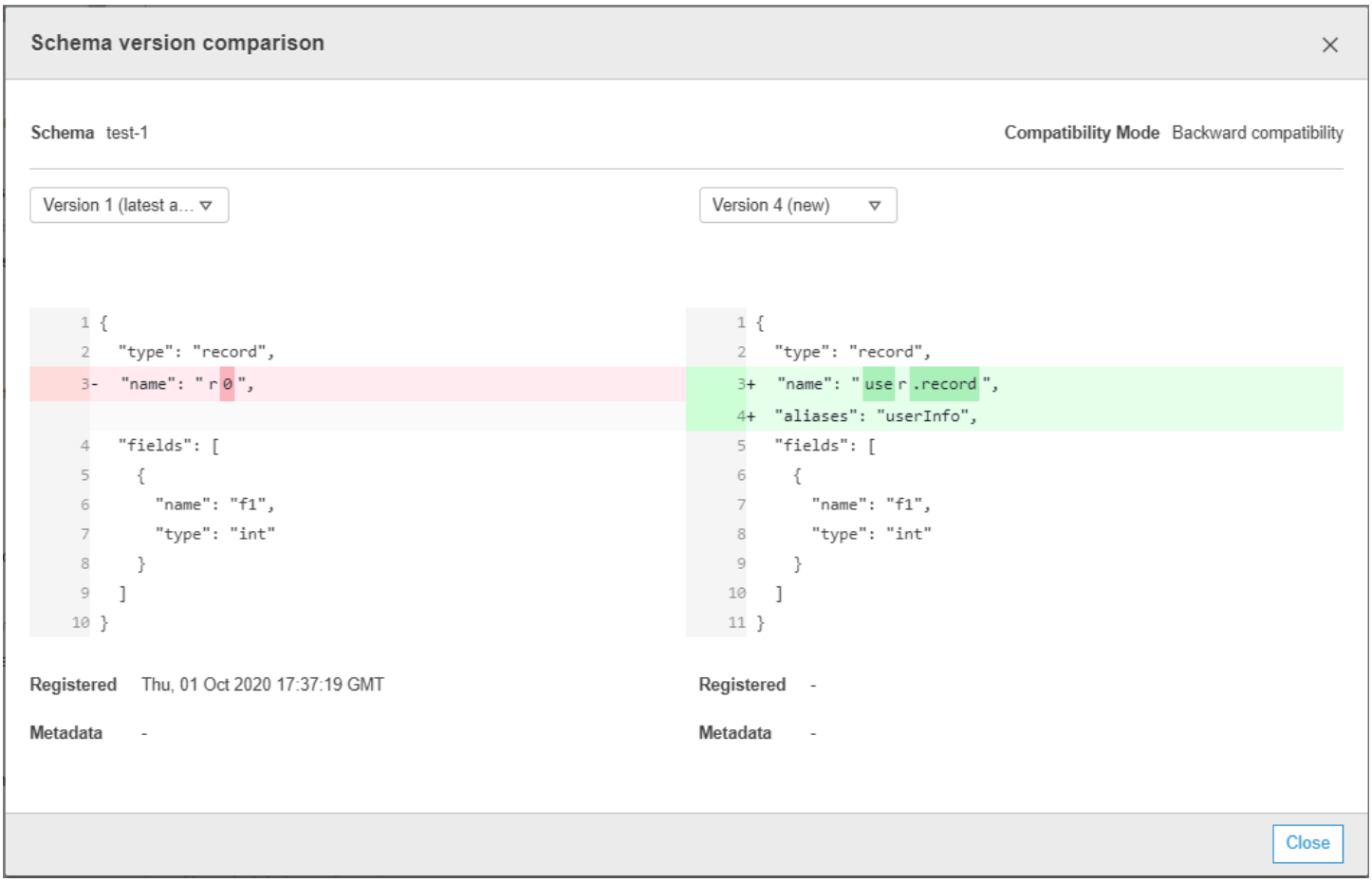
La versión de esquema(s) aparece en la lista de versiones. Si la versión cambió el modo de compatibilidad, la versión se marcará como punto de control.

Ejemplo de comparación de versiones de esquema.

Cuando elige Compare with previous version (Comparar con la versión anterior), verá las versiones anterior y nueva mostradas juntas. La información que se modificó se resaltará de la siguiente manera:

- Amarillo: indica información modificada.
- Verde: indica el contenido agregado en la última versión.
- Rojo: indica el contenido eliminado de la última versión.

También se pueden realizar comparaciones con versiones anteriores.



Eliminación de un esquema o registro

Eliminar un esquema, una versión de esquema o un registro son acciones permanentes que no se pueden deshacer.

Eliminación de un esquema

Es posible que desee eliminar un esquema cuando ya no se utilizará dentro de un registro, utilizando la consola AWS Management Console, o la API DeleteSchema acción (Python: delete_schema).

Eliminar uno o varios esquemas es una acción permanente que no se puede deshacer. Asegúrese de que el esquema o los esquemas ya no son necesarios.

Para eliminar un esquema del registro, llame a la API DeleteSchema acción (Python: delete_schema), y especifique la estructura de `SchemaId` para identificar el esquema.

Por ejemplo:

```
aws glue delete-schema --schema-id SchemaArn="arn:aws:glue:us-
east-2:901234567890:schema/registryName1/schemaname"
```

```
aws glue delete-schema --schema-id SchemaName="TestSchema6-
deleteschemabyname",RegistryName="default-registry"
```

Consola de AWS Glue

Para eliminar un esquema de la consola de AWS Glue:

1. Inicie sesión en AWS Management Console y abra la consola de AWS Glue en https://console.aws.amazon.com/glue/.
2. En el panel de navegación, en Data catalog (Catálogo de datos), elija Schema registries (Registros de esquemas).
3. Elija el registro que contiene el esquema de la lista de registros.
4. Seleccione uno o varios esquemas de la lista, al marcar las casillas correspondientes.
5. En el menú Action (Acción), elija Delete schema (Eliminar esquema).
6. Ingrese el texto **`Delete`** en el campo para confirmar la eliminación.
7. Elija Eliminar.

Los esquemas especificados se eliminan del registro.

Eliminar una versión de esquema

A medida que los esquemas se acumulan en el registro, es posible que desee eliminar las versiones de esquema no deseadas mediante la AWS Management Console o la API DeleteSchemaVersions acción (Python: delete_schema_versions). Eliminar una o varias versiones de esquema es una acción permanente que no se puede deshacer. Asegúrese de que las versiones del esquema ya no sean necesarias.

Al eliminar versiones de esquema, tenga en cuenta las siguientes restricciones:

- No puede eliminar una versión marcada como punto de control.

- El rango de versiones contiguas no puede ser superior a 25.
- La versión del esquema más reciente no debe estar en un estado pendiente.

Especifique la estructura de `SchemaId` para identificar el esquema y especifique las `Versions` como un rango de versiones para eliminar. Para obtener más información sobre cómo especificar una versión o un rango de versiones, consulte DeleteRegistry acción (Python: delete_registry). Las versiones de los esquemas especificados se eliminan del registro.

Llamar a la API ListSchemaVersions acción (Python: list_schema_versions) después de esta llamada arrojará una lista del estado de las versiones eliminadas.

Por ejemplo:

```
aws glue delete-schema-versions --schema-id
 SchemaName="TestSchema6",RegistryName="default-registry" --versions "1-1"
```

```
aws glue delete-schema-versions --schema-id SchemaArn="arn:aws:glue:us-
east-2:901234567890:schema/default-registry/TestSchema6-NON-Existent" --versions "1-1"
```

1. Inicie sesión en AWS Management Console y abra la consola de AWS Glue en https://console.aws.amazon.com/glue/.
2. En el panel de navegación, en Data catalog (Catálogo de datos), elija Schema registries (Registros de esquemas).
3. Elija el registro que contiene el esquema de la lista de registros.
4. Seleccione uno o varios esquemas de la lista, al marcar las casillas correspondientes.
5. En el menú Action (Acción), elija Delete schema (Eliminar esquema).
6. Ingrese el texto **`Delete`** en el campo para confirmar la eliminación.
7. Elija Eliminar.

Las versiones de los esquemas especificados se eliminan del registro.

Eliminación de un registro

Es posible que desee eliminar un registro cuando los esquemas que contiene ya no se organicen en ese registro. Tendrá que reasignar esos esquemas a otro registro.

Eliminar uno o varios registros es una acción permanente que no se puede deshacer. Asegúrese de que el registro o los registros ya no son necesarios.

El registro predeterminado se puede eliminar mediante AWS CLI.

API de AWS Glue

Para eliminar todo el registro, incluido el esquema y todas sus versiones, llame a la API DeleteRegistry acción (Python: delete_registry). Especifique una estructura de `RegistryId` para identificar el registro.

Por ejemplo:

```
aws glue delete-registry --registry-id RegistryArn="arn:aws:glue:us-
east-2:901234567890:registry/registryName1"
```

```
aws glue delete-registry --registry-id RegistryName="TestRegistry-deletebyname"
```

Para obtener el estado de la operación de eliminación, puede llamar a la API `GetRegistry` después de la llamada asíncrona.

Consola de AWS Glue

Para eliminar un registro de la consola de AWS Glue:

1. Inicie sesión en AWS Management Console y abra la consola de AWS Glue en https://console.aws.amazon.com/glue/.
2. En el panel de navegación, en Data catalog (Catálogo de datos), elija Schema registries (Registros de esquemas).
3. Seleccione un registro de la lista, al marcar la casilla correspondiente.
4. En el menú Action (Acción), elija Delete registry (Eliminar registro).
5. Ingrese el texto **`Delete`** en el campo para confirmar la eliminación.
6. Elija Eliminar.

Los registros seleccionados se eliminan de AWS Glue.

Ejemplos de IAM para serializadores

> **Note**
>
> Las políticas administradas de AWS conceden los permisos necesarios para los casos de uso comunes. Para obtener información sobre el uso de las políticas para administrar el registro de esquemas, consulte AWS políticas gestionadas (predefinidas) para AWS Glue.

Para los serializadores, debe crear una política mínima similar a la siguiente para poder encontrar el `schemaVersionId` para una definición de esquema determinada. Tenga en cuenta que debe tener permisos de lectura en el registro para poder leer los esquemas del registro. Puede limitar los registros que se pueden leer mediante la cláusula `Resource`.

Ejemplos de código 13:

```
{
    "Sid" : "GetSchemaByDefinition",
    "Effect" : "Allow",
    "Action" :
 [
        "glue:GetSchemaByDefinition"
    ],
        "Resource" : ["arn:aws:glue:us-east-2:012345678:registry/registryname-1",
                      "arn:aws:glue:us-east-2:012345678:schema/registryname-1/
schemaname-1",
                      "arn:aws:glue:us-east-2:012345678:schema/registryname-1/
schemaname-2"
                     ]
}
```

Además, también puede permitir a los productores crear nuevos esquemas y versiones mediante la inclusión de los siguientes métodos adicionales. Tenga en cuenta que debería poder inspeccionar el registro para agregar/eliminar/evolucionar los esquemas dentro de él. Puede limitar los registros que se pueden inspeccionar mediante la cláusula `Resource`.

Ejemplos de código 14:

```
{
    "Sid" : "RegisterSchemaWithMetadata",
```

```
    "Effect" : "Allow",
    "Action" :
 [
        "glue:GetSchemaByDefinition",
        "glue:CreateSchema",
        "glue:RegisterSchemaVersion",
        "glue:PutSchemaVersionMetadata",
    ],
    "Resource" : ["arn:aws:glue:aws-region:123456789012:registry/registryname-1",
                  "arn:aws:glue:aws-region:123456789012:schema/registryname-1/
schemaname-1",
                  "arn:aws:glue:aws-region:123456789012:schema/registryname-1/
schemaname-2"
                ]
}
```

Ejemplos de IAM para deserializadores

Para los deserializadores (lado del consumidor), debe crear una política similar a la siguiente para permitir que el deserializador recupere el esquema de Schema Registry para su deserialización. Tenga en cuenta que debería poder inspeccionar el registro para obtener los esquemas dentro de él.

Ejemplos de código 15:

```
{
    "Sid" : "GetSchemaVersion",
    "Effect" : "Allow",
    "Action" :
 [
        "glue:GetSchemaVersion"
    ],
    "Resource" : ["*"]
}
```

Conectividad privada mediante AWS PrivateLink

Puede usar AWS PrivateLink para conectar la VPC del productor de datos a AWS Glue. Para ello, defina un punto de enlace de la VPC de interfaz para AWS Glue. Cuando se utiliza un punto de enlace de la interfaz de la VPC, la comunicación entre la VPC y AWS Glue se realiza en su totalidad dentro de la red de AWS. Para obtener más información, consulte Uso de AWS Glue con puntos de enlace de la VPC.

Acceso a métricas de Amazon CloudWatch

Las métricas de Amazon CloudWatch están disponibles como parte de la capa gratuita de CloudWatch. También puede acceder a estas métricas en la consola de CloudWatch. Las métricas de nivel de API incluyen CreateSchema (éxito y latencia), GetSchemabyDefinition, (éxito y latencia), GetSchemaVersion (éxito y latencia), RegisterSchemaVersion (éxito y latencia), PutSchemaVersionMetadata (éxito y latencia). Las métricas de nivel de recursos incluyen Registry.ThrottledByLimit, SchemaVersion.ThrottledByLimit, SchemaVersion.Size.

Ejemplo de plantilla de AWS CloudFormation para Schema Registry

A continuación se muestra una plantilla de ejemplo para crear recursos de Schema Registry en AWS CloudFormation. Para crear esta pila en su cuenta, copie la plantilla anterior en un archivo `SampleTemplate.yaml` y ejecute el siguiente comando:

```
aws cloudformation create-stack --stack-name ABCSchemaRegistryStack --template-body
 "'cat SampleTemplate.yaml'"
```

En este ejemplo se utiliza `AWS::Glue::Registry` para crear un registro, `AWS::Glue::Schema` para crear un esquema, `AWS::Glue::SchemaVersion` para crear una versión de esquema y `AWS::Glue::SchemaVersionMetadata` para completar los metadatos de la versión del esquema.

```
Description: "A sample CloudFormation template for creating Schema Registry resources."
Resources:
  ABCRegistry:
    Type: "AWS::Glue::Registry"
    Properties:
      Name: "ABCSchemaRegistry"
      Description: "ABC Corp. Schema Registry"
      Tags:
        - Key: "Project"
          Value: "Foo"
  ABCSchema:
    Type: "AWS::Glue::Schema"
    Properties:
      Registry:
        Arn: !Ref ABCRegistry
      Name: "TestSchema"
      Compatibility: "NONE"
      DataFormat: "AVRO"
      SchemaDefinition: >
```

```
        {"namespace":"foo.avro","type":"record","name":"user","fields":
[{"name":"name","type":"string"},{"name":"favorite_number","type":"int"}]}
      Tags:
        - Key: "Project"
          Value: "Foo"
  SecondSchemaVersion:
    Type: "AWS::Glue::SchemaVersion"
    Properties:
      Schema:
        SchemaArn: !Ref ABCSchema
      SchemaDefinition: >
        {"namespace":"foo.avro","type":"record","name":"user","fields":
[{"name":"status","type":"string", "default":"ON"}, {"name":"name","type":"string"},
{"name":"favorite_number","type":"int"}]}
  FirstSchemaVersionMetadata:
    Type: "AWS::Glue::SchemaVersionMetadata"
    Properties:
      SchemaVersionId: !GetAtt ABCSchema.InitialSchemaVersionId
      Key: "Application"
      Value: "Kinesis"
  SecondSchemaVersionMetadata:
    Type: "AWS::Glue::SchemaVersionMetadata"
    Properties:
      SchemaVersionId: !Ref SecondSchemaVersion
      Key: "Application"
      Value: "Kinesis"
```

Integración con AWS Glue Schema Registry

En estas secciones se describen las integraciones con AWS Glue Schema Registry. Los ejemplos de esta sección muestran un esquema con formato de datos AVRO. Para obtener más ejemplos, incluidos esquemas con formato de datos JSON, consulte las pruebas de integración y la información de ReadMe (Léame) en el Repositorio de código abierto de AWS Glue Schema Registry.

Temas

- Caso de uso: Conexión de Schema Registry a Amazon MSK o Apache Kafka
- Caso de uso: Integración de Amazon Kinesis Data Streams con AWS Glue Schema Registry
- Caso de uso: Amazon Managed Service para Apache Flink
- Caso de uso: integración con AWS Lambda

- Caso de uso: AWS Glue Data Catalog
- Caso de uso: Streaming de AWS Glue
- Caso de uso: Apache Kafka Streams
- Caso de uso: Apache Kafka Connect

Caso de uso: Conexión de Schema Registry a Amazon MSK o Apache Kafka

Supongamos que está escribiendo datos en un tema de Apache Kafka. Puede seguir estos pasos para comenzar.

1. Cree un clúster de Amazon Managed Streaming for Apache Kafka (Amazon MSK) o Apache Kafka con al menos un tema. Si crea un clúster de Amazon MSK, puede utilizar la AWS Management Console. Siga las siguientes isntrucciones: Introducción al uso de Amazon MSK en la Guía para desarrolladores de Amazon Managed Streaming for Apache Kafka.
2. Siga el paso Instalación de bibliotecas SerDe anterior.
3. Para crear registros de esquema, esquemas o versiones de esquema, siga las instrucciones de la sección Introducción a Schema Registry de este documento.
4. Inicie a sus productores y consumidores en el uso de Schema Registry para escribir y leer registros a/desde el tema de Amazon MSK o Apache Kafka. Puede encontrar un ejemplo de código de productor y consumidor en el archivo ReadMe (Léame) de las bibliotecas Serde. La biblioteca de Schema Registry del productor serializará automáticamente el registro y agregará un ID de versión de esquema al registro.
5. Si se ha introducido el esquema de este registro, o si el registro automático está activado, el esquema se habrá registrado en Schema Registry.
6. El consumidor que lee el tema de Amazon MSK o Apache Kafka, con la biblioteca de AWS Glue Schema Registry, buscará automáticamente el esquema desde Schema Registry.

Caso de uso: Integración de Amazon Kinesis Data Streams con AWS Glue Schema Registry

Esta integración requiere que tenga un flujo de datos de Amazon Kinesis. Para obtener más información, consulte Introducción a Amazon Kinesis Data Streams en la Guía para desarrolladores de Amazon Kinesis Data Streams.

Existen dos formas de interactuar con los datos en un flujo de datos de Kinesis.

- A través de las bibliotecas Kinesis Producer Library (KPL) y Kinesis Client Library (KCL) en Java. No se proporciona soporte multilingüe.
- A través de las API `PutRecords`, `PutRecord` y `GetRecords` de Kinesis Data Streams disponibles en AWS SDK for Java.

Si utiliza actualmente las bibliotecas KPL/KCL, le recomendamos seguir utilizando ese método. Hay versiones actualizadas de KCL y KPL con Schema Registry integrado, como se muestra en los ejemplos. De lo contrario, puede utilizar el código de muestra para aprovechar el AWS Glue Schema Registry si utiliza las API de KDS directamente.

La integración de Schema Registry sólo está disponible con KPL v0.14.2 o posterior y con KCL v2.3 o posterior. La integración de Schema Registry con datos JSON sólo está disponible con KPL v0.14.8 o posterior y con KCL v2.3.6 o posterior.

Interacción con datos mediante SDK de Kinesis V2

En esta sección se describe la interacción con Kinesis mediante SDK de Kinesis V2

```
// Example JSON Record, you can construct a AVRO record also
private static final JsonDataWithSchema record =
 JsonDataWithSchema.builder(schemaString, payloadString);
private static final DataFormat dataFormat = DataFormat.JSON;

//Configurations for Schema Registry
GlueSchemaRegistryConfiguration gsrConfig = new GlueSchemaRegistryConfiguration("us-
east-1");

GlueSchemaRegistrySerializer glueSchemaRegistrySerializer =
        new GlueSchemaRegistrySerializerImpl(awsCredentialsProvider, gsrConfig);
GlueSchemaRegistryDataFormatSerializer dataFormatSerializer =
        new GlueSchemaRegistrySerializerFactory().getInstance(dataFormat, gsrConfig);

Schema gsrSchema =
        new Schema(dataFormatSerializer.getSchemaDefinition(record), dataFormat.name(),
 "MySchema");

byte[] serializedBytes = dataFormatSerializer.serialize(record);

byte[] gsrEncodedBytes = glueSchemaRegistrySerializer.encode(streamName, gsrSchema,
 serializedBytes);
```

```
PutRecordRequest putRecordRequest = PutRecordRequest.builder()
        .streamName(streamName)
        .partitionKey("partitionKey")
        .data(SdkBytes.fromByteArray(gsrEncodedBytes))
        .build();
shardId = kinesisClient.putRecord(putRecordRequest)
        .get()
        .shardId();

GlueSchemaRegistryDeserializer glueSchemaRegistryDeserializer = new
 GlueSchemaRegistryDeserializerImpl(awsCredentialsProvider, gsrConfig);

GlueSchemaRegistryDataFormatDeserializer gsrDataFormatDeserializer =
        glueSchemaRegistryDeserializerFactory.getInstance(dataFormat, gsrConfig);

GetShardIteratorRequest getShardIteratorRequest = GetShardIteratorRequest.builder()
        .streamName(streamName)
        .shardId(shardId)
        .shardIteratorType(ShardIteratorType.TRIM_HORIZON)
        .build();

String shardIterator = kinesisClient.getShardIterator(getShardIteratorRequest)
        .get()
        .shardIterator();

GetRecordsRequest getRecordRequest = GetRecordsRequest.builder()
        .shardIterator(shardIterator)
        .build();
GetRecordsResponse recordsResponse = kinesisClient.getRecords(getRecordRequest)
        .get();

List<Object> consumerRecords = new ArrayList<>();
List<Record> recordsFromKinesis = recordsResponse.records();

for (int i = 0; i < recordsFromKinesis.size(); i++) {
    byte[] consumedBytes = recordsFromKinesis.get(i)
             .data()
             .asByteArray();

    Schema gsrSchema = glueSchemaRegistryDeserializer.getSchema(consumedBytes);
    Object decodedRecord =
 gsrDataFormatDeserializer.deserialize(ByteBuffer.wrap(consumedBytes),

 gsrSchema.getSchemaDefinition());
```

```
    consumerRecords.add(decodedRecord);
}
```

Interacción con los datos mediante las bibliotecas KPL/KCL

En esta sección se describe la integración de Kinesis Data Streams con Schema Registry mediante las bibliotecas KPL/KCL. Para obtener más información sobre el uso de KPL/KCL, consulte Desarrollar productores con Amazon Kinesis Producer Library en la Guía para desarrolladores de Amazon Kinesis Data Streams.

Configuración de Schema Registry en KPL

1. Establezca la definición de esquema para los datos, el formato de datos y el nombre del esquema creados en AWS Glue Schema Registry.
2. Opcionalmente, configure el objeto `GlueSchemaRegistryConfiguration`.
3. Transfiera el objeto de esquema a `addUserRecord API`.

```
private static final String SCHEMA_DEFINITION = "{"namespace": "example.avro",\n"
+ " "type": "record",\n"
+ " "name": "User",\n"
+ " "fields": [\n"
+ " {"name": "name", "type": "string"},\n"
+ " {"name": "favorite_number", "type": ["int", "null"]},\n"
+ " {"name": "favorite_color", "type": ["string", "null"]}\n"
+ " ]\n"
+ "}";

KinesisProducerConfiguration config = new KinesisProducerConfiguration();
config.setRegion("us-west-1")

//[Optional] configuration for Schema Registry.

GlueSchemaRegistryConfiguration schemaRegistryConfig =
new GlueSchemaRegistryConfiguration("us-west-1");

schemaRegistryConfig.setCompression(true);

config.setGlueSchemaRegistryConfiguration(schemaRegistryConfig);

///Optional configuration ends.

final KinesisProducer producer =
```

```
      new KinesisProducer(config);

final ByteBuffer data = getDataToSend();

com.amazonaws.services.schemaregistry.common.Schema gsrSchema =
    new Schema(SCHEMA_DEFINITION, DataFormat.AVRO.toString(), "demoSchema");

ListenableFuture<UserRecordResult> f = producer.addUserRecord(
config.getStreamName(), TIMESTAMP, Utils.randomExplicitHashKey(), data, gsrSchema);

private static ByteBuffer getDataToSend() {
      org.apache.avro.Schema avroSchema =
        new org.apache.avro.Schema.Parser().parse(SCHEMA_DEFINITION);

      GenericRecord user = new GenericData.Record(avroSchema);
      user.put("name", "Emily");
      user.put("favorite_number", 32);
      user.put("favorite_color", "green");

      ByteArrayOutputStream outBytes = new ByteArrayOutputStream();
      Encoder encoder = EncoderFactory.get().directBinaryEncoder(outBytes, null);
      new GenericDatumWriter<>(avroSchema).write(user, encoder);
      encoder.flush();
      return ByteBuffer.wrap(outBytes.toByteArray());
 }
```

Configuración de Kinesis Client Library

Desarrolle un consumidor de Kinesis Client Library en Java. Para obtener más información, consulte Desarrollo de un consumidor de Kinesis Client Library en Java en la Guía para desarrolladores de Amazon Kinesis Data Streams.

1. Cree una instancia de `GlueSchemaRegistryDeserializer` al transferir un objeto `GlueSchemaRegistryConfiguration`.
2. Transfiera el `GlueSchemaRegistryDeserializer` a `retrievalConfig.glueSchemaRegistryDeserializer`.
3. Acceda al esquema de los mensajes entrantes al llamar a `kinesisClientRecord.getSchema()`.

```
GlueSchemaRegistryConfiguration schemaRegistryConfig =
    new GlueSchemaRegistryConfiguration(this.region.toString());
```

```
GlueSchemaRegistryDeserializer glueSchemaRegistryDeserializer =
   new
GlueSchemaRegistryDeserializerImpl(DefaultCredentialsProvider.builder().build(),
schemaRegistryConfig);

RetrievalConfig retrievalConfig =
configsBuilder.retrievalConfig().retrievalSpecificConfig(new
PollingConfig(streamName, kinesisClient));
retrievalConfig.glueSchemaRegistryDeserializer(glueSchemaRegistryDeserializer);

 Scheduler scheduler = new Scheduler(
           configsBuilder.checkpointConfig(),
           configsBuilder.coordinatorConfig(),
           configsBuilder.leaseManagementConfig(),
           configsBuilder.lifecycleConfig(),
           configsBuilder.metricsConfig(),
           configsBuilder.processorConfig(),
           retrievalConfig
       );

public void processRecords(ProcessRecordsInput processRecordsInput) {
           MDC.put(SHARD_ID_MDC_KEY, shardId);
           try {
               log.info("Processing {} record(s)",
               processRecordsInput.records().size());
               processRecordsInput.records()
               .forEach(
                   r ->
                       log.info("Processed record pk: {} -- Seq: {} : data {} with
schema: {}",
                       r.partitionKey(),
r.sequenceNumber(), recordToAvroObj(r).toString(), r.getSchema()));
           } catch (Throwable t) {
               log.error("Caught throwable while processing records. Aborting.");
               Runtime.getRuntime().halt(1);
           } finally {
               MDC.remove(SHARD_ID_MDC_KEY);
           }
}

private GenericRecord recordToAvroObj(KinesisClientRecord r) {
   byte[] data = new byte[r.data().remaining()];
   r.data().get(data, 0, data.length);
```

```
    org.apache.avro.Schema schema = new
 org.apache.avro.Schema.Parser().parse(r.schema().getSchemaDefinition());
    DatumReader datumReader = new GenericDatumReader<>(schema);

    BinaryDecoder binaryDecoder = DecoderFactory.get().binaryDecoder(data, 0,
 data.length, null);
    return (GenericRecord) datumReader.read(null, binaryDecoder);
 }
```

Interacción con datos mediante las API de Kinesis Data Streams

En esta sección se describe la integración de Kinesis Data Streams con Schema Registry mediante las API de Kinesis Data Streams.

1. Actualice estas dependencias de Maven:

```
<dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>com.amazonaws</groupId>
                <artifactId>aws-java-sdk-bom</artifactId>
                <version>1.11.884</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
        </dependencies>
    </dependencyManagement>

    <dependencies>
        <dependency>
            <groupId>com.amazonaws</groupId>
            <artifactId>aws-java-sdk-kinesis</artifactId>
        </dependency>

        <dependency>
            <groupId>software.amazon.glue</groupId>
            <artifactId>schema-registry-serde</artifactId>
            <version>1.1.5</version>
        </dependency>

        <dependency>
            <groupId>com.fasterxml.jackson.dataformat</groupId>
```

```
            <artifactId>jackson-dataformat-cbor</artifactId>
            <version>2.11.3</version>
        </dependency>
    </dependencies>
```

2. En el productor, agregue información de encabezado de esquema con la API `PutRecords` o `PutRecord` en Kinesis Data Streams.

```
//The following lines add a Schema Header to the record
        com.amazonaws.services.schemaregistry.common.Schema awsSchema =
            new com.amazonaws.services.schemaregistry.common.Schema(schemaDefinition,
 DataFormat.AVRO.name(),
                schemaName);
        GlueSchemaRegistrySerializerImpl glueSchemaRegistrySerializer =
            new
 GlueSchemaRegistrySerializerImpl(DefaultCredentialsProvider.builder().build(), new
 GlueSchemaRegistryConfiguration(getConfigs()));
        byte[] recordWithSchemaHeader =
            glueSchemaRegistrySerializer.encode(streamName, awsSchema,
 recordAsBytes);
```

3. En el productor, use la API `PutRecords` o `PutRecord` para poner el registro en el flujo de datos.
4. En el consumidor, elimine el registro de esquema del encabezado y serialice un registro de esquemas de Avro.

```
//The following lines remove Schema Header from record
        GlueSchemaRegistryDeserializerImpl glueSchemaRegistryDeserializer =
            new
 GlueSchemaRegistryDeserializerImpl(DefaultCredentialsProvider.builder().build(),
 getConfigs());
        byte[] recordWithSchemaHeaderBytes = new
 byte[recordWithSchemaHeader.remaining()];
        recordWithSchemaHeader.get(recordWithSchemaHeaderBytes, 0,
 recordWithSchemaHeaderBytes.length);
        com.amazonaws.services.schemaregistry.common.Schema awsSchema =
            glueSchemaRegistryDeserializer.getSchema(recordWithSchemaHeaderBytes);
        byte[] record =
 glueSchemaRegistryDeserializer.getData(recordWithSchemaHeaderBytes);

        //The following lines serialize an AVRO schema record
        if (DataFormat.AVRO.name().equals(awsSchema.getDataFormat())) {
            Schema avroSchema = new
 org.apache.avro.Schema.Parser().parse(awsSchema.getSchemaDefinition());
```

```
            Object genericRecord = convertBytesToRecord(avroSchema, record);
            System.out.println(genericRecord);
        }
```

Interacción con datos mediante las API de Kinesis Data Streams

El siguiente es el código de ejemplo para usar las API `PutRecords` y `GetRecords`.

```
//Full sample code
import
 com.amazonaws.services.schemaregistry.deserializers.GlueSchemaRegistryDeserializerImpl;
import
 com.amazonaws.services.schemaregistry.serializers.GlueSchemaRegistrySerializerImpl;
import com.amazonaws.services.schemaregistry.utils.AVROUtils;
import com.amazonaws.services.schemaregistry.utils.AWSSchemaRegistryConstants;
import org.apache.avro.Schema;
import org.apache.avro.generic.GenericData;
import org.apache.avro.generic.GenericDatumReader;
import org.apache.avro.generic.GenericDatumWriter;
import org.apache.avro.generic.GenericRecord;
import org.apache.avro.io.Decoder;
import org.apache.avro.io.DecoderFactory;
import org.apache.avro.io.Encoder;
import org.apache.avro.io.EncoderFactory;
import software.amazon.awssdk.auth.credentials.DefaultCredentialsProvider;
import software.amazon.awssdk.services.glue.model.DataFormat;

import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.IOException;
import java.nio.ByteBuffer;
import java.util.Collections;
import java.util.HashMap;
import java.util.Map;


public class PutAndGetExampleWithEncodedData {
    static final String regionName = "us-east-2";
    static final String streamName = "testStream1";
    static final String schemaName = "User-Topic";
    static final String AVRO_USER_SCHEMA_FILE = "src/main/resources/user.avsc";
    KinesisApi kinesisApi = new KinesisApi();
```

```
   void runSampleForPutRecord() throws IOException {
       Object testRecord = getTestRecord();
       byte[] recordAsBytes = convertRecordToBytes(testRecord);
       String schemaDefinition =
AVROUtils.getInstance().getSchemaDefinition(testRecord);

       //The following lines add a Schema Header to a record
       com.amazonaws.services.schemaregistry.common.Schema awsSchema =
           new com.amazonaws.services.schemaregistry.common.Schema(schemaDefinition,
DataFormat.AVRO.name(),
               schemaName);
       GlueSchemaRegistrySerializerImpl glueSchemaRegistrySerializer =
           new
GlueSchemaRegistrySerializerImpl(DefaultCredentialsProvider.builder().build(), new
GlueSchemaRegistryConfiguration(regionName));
       byte[] recordWithSchemaHeader =
           glueSchemaRegistrySerializer.encode(streamName, awsSchema, recordAsBytes);

       //Use PutRecords api to pass a list of records
       kinesisApi.putRecords(Collections.singletonList(recordWithSchemaHeader),
streamName, regionName);

       //OR
       //Use PutRecord api to pass single record
       //kinesisApi.putRecord(recordWithSchemaHeader, streamName, regionName);
   }

   byte[] runSampleForGetRecord() throws IOException {
       ByteBuffer recordWithSchemaHeader = kinesisApi.getRecords(streamName,
regionName);

       //The following lines remove the schema registry header
       GlueSchemaRegistryDeserializerImpl glueSchemaRegistryDeserializer =
           new
GlueSchemaRegistryDeserializerImpl(DefaultCredentialsProvider.builder().build(), new
GlueSchemaRegistryConfiguration(regionName));
       byte[] recordWithSchemaHeaderBytes = new
byte[recordWithSchemaHeader.remaining()];
       recordWithSchemaHeader.get(recordWithSchemaHeaderBytes, 0,
recordWithSchemaHeaderBytes.length);

       com.amazonaws.services.schemaregistry.common.Schema awsSchema =
           glueSchemaRegistryDeserializer.getSchema(recordWithSchemaHeaderBytes);
```

```
        byte[] record =
glueSchemaRegistryDeserializer.getData(recordWithSchemaHeaderBytes);

        //The following lines serialize an AVRO schema record
        if (DataFormat.AVRO.name().equals(awsSchema.getDataFormat())) {
            Schema avroSchema = new
org.apache.avro.Schema.Parser().parse(awsSchema.getSchemaDefinition());
            Object genericRecord = convertBytesToRecord(avroSchema, record);
            System.out.println(genericRecord);
        }

        return record;
    }

    private byte[] convertRecordToBytes(final Object record) throws IOException {
        ByteArrayOutputStream recordAsBytes = new ByteArrayOutputStream();
        Encoder encoder = EncoderFactory.get().directBinaryEncoder(recordAsBytes,
null);
        GenericDatumWriter datumWriter = new
GenericDatumWriter<>(AVROUtils.getInstance().getSchema(record));
        datumWriter.write(record, encoder);
        encoder.flush();
        return recordAsBytes.toByteArray();
    }

    private GenericRecord convertBytesToRecord(Schema avroSchema, byte[] record) throws
IOException {
        final GenericDatumReader<GenericRecord> datumReader = new
GenericDatumReader<>(avroSchema);
        Decoder decoder = DecoderFactory.get().binaryDecoder(record, null);
        GenericRecord genericRecord = datumReader.read(null, decoder);
        return genericRecord;
    }

    private Map<String, String> getMetadata() {
        Map<String, String> metadata = new HashMap<>();
        metadata.put("event-source-1", "topic1");
        metadata.put("event-source-2", "topic2");
        metadata.put("event-source-3", "topic3");
        metadata.put("event-source-4", "topic4");
        metadata.put("event-source-5", "topic5");
        return metadata;
    }
```

```
    private GlueSchemaRegistryConfiguration getConfigs() {
        GlueSchemaRegistryConfiguration configs = new
 GlueSchemaRegistryConfiguration(regionName);
        configs.setSchemaName(schemaName);
        configs.setAutoRegistration(true);
        configs.setMetadata(getMetadata());
        return configs;
    }

    private Object getTestRecord() throws IOException {
        GenericRecord genericRecord;
        Schema.Parser parser = new Schema.Parser();
        Schema avroSchema = parser.parse(new File(AVRO_USER_SCHEMA_FILE));

        genericRecord = new GenericData.Record(avroSchema);
        genericRecord.put("name", "testName");
        genericRecord.put("favorite_number", 99);
        genericRecord.put("favorite_color", "red");

        return genericRecord;
    }
}
```

Caso de uso: Amazon Managed Service para Apache Flink

Apache Flink es un marco de código abierto y motor de procesamiento distribuido popular para informática con estado sobre flujos de datos ilimitados y delimitados. Amazon Managed Service para Apache Flink es un servicio de AWS completamente administrado que permite crear y administrar aplicaciones de Apache Flink para procesar datos de streaming.

El código abierto Apache Flink proporciona una serie de orígenes y receptores. Por ejemplo, los orígenes de datos predefinidos incluyen la lectura de archivos, directorios y sockets, y la ingesta de datos de recopilaciones e iteradores. Los conectores Apache Flink DataStream proporcionan código para que Apache Flink interactúe con varios sistemas de terceros, como Apache Kafka o Kinesis como orígenes o receptores.

Para obtener más información, consulte la Guía para desarrolladores de Amazon Kinesis Data Analytics.

Conector Kafka de Apache Flink

Apache Flink proporciona un conector de flujo de datos Apache Kafka para leer y escribir datos en temas de Kafka con garantías de una sola vez. El consumidor Kafka de Flink, `FlinkKafkaConsumer`, proporciona acceso a la lectura de uno o más temas de Kafka. El productor Kafka de Apache Flink, `FlinkKafkaProducer`, permite escribir una secuencia de registros en uno o más temas de Kafka. Para obtener más información, consulte Conector de Apache Kafka.

Conector de flujos de Kinesis de Apache Flink

El conector de flujo de datos de Kinesis proporciona acceso a Amazon Kinesis Data Streams. El `FlinkKinesisConsumer` es un origen de datos de streaming en paralelo de exactamente una única vez que se suscribe a múltiples flujos de Kinesis dentro de la misma región de servicio de AWS, y puede manejar de forma transparente la redistribución de flujos mientras el trabajo se está ejecutando. Cada subtarea del consumidor es responsable de obtener registros de datos de múltiples fragmentos de Kinesis. El número de fragmentos obtenidos por cada subtarea cambiará a medida que Kinesis cierre y cree fragmentos. El `FlinkKinesisProducer` utiliza Kinesis Producer Library (KPL) para poner los datos de un flujo de Apache Flink en un flujo de Kinesis. Para obtener más información, consulte Conector de Amazon Kinesis Streams.

Para obtener más información, consulte el repositorio GitHub de esquemas de AWS Glue.

Integración con Apache Flink

La biblioteca SerDes proporcionada con Schema Registry se integra con Apache Flink. Para trabajar con Apache Flink, debe implementar las interfaces de `SerializationSchema` y `DeserializationSchema`, denominadas `GlueSchemaRegistryAvroSerializationSchema` y `GlueSchemaRegistryAvroDeserializationSchema`, que puede conectar a los conectores Apache Flink.

Adición de una dependencia de AWS Glue Schema Registry en la aplicación Apache Flink

Para configurar las dependencias de integración a AWS Glue Schema Registry en la aplicación Apache Flink:

1. Agregue la dependencia al archivo `pom.xml`.

```
<dependency>
    <groupId>software.amazon.glue</groupId>
    <artifactId>schema-registry-flink-serde</artifactId>
    <version>1.0.0</version>
```

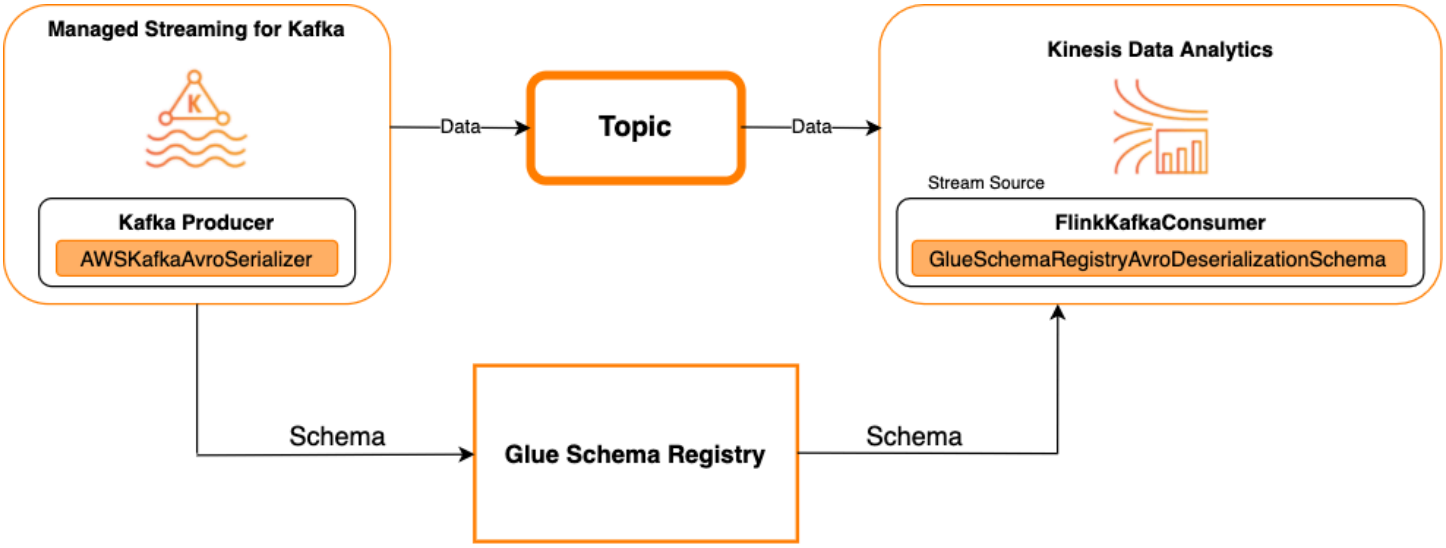
```
</dependency>
```

Integración de Kafka o Amazon MSK con Apache Flink

Puede usar Managed Service para Apache Flink con Kafka como origen o receptor.

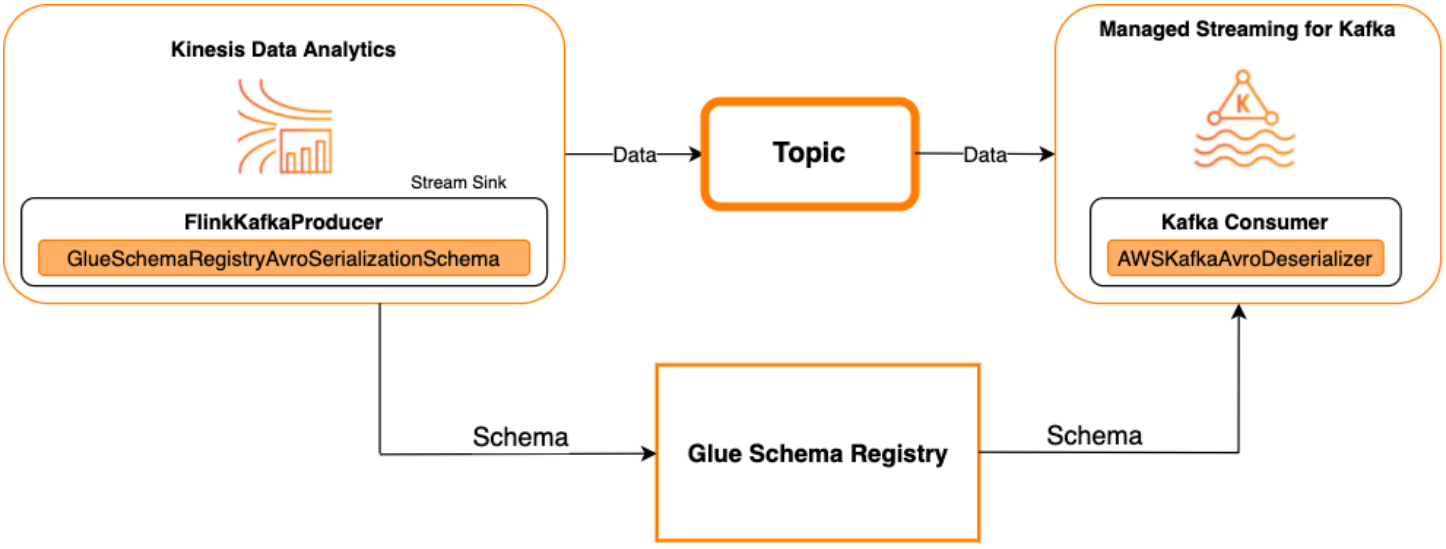
Kafka como origen

En el siguiente diagrama, se muestra la integración de Kinesis Data Streams con Managed Service para Apache Flink, con Kafka como origen.



Kafka como receptor

En el siguiente diagrama, se muestra la integración de Kinesis Data Streams con Managed Service para Apache Flink, con Kafka como receptor.



Para integrar Kafka (o Amazon MSK) con Managed Service para Apache Flink, con Kafka como origen o receptor, realice los siguientes cambios de código. Agregue los bloques de código en negrita a su código respectivo en las secciones análogas.

Si Kafka es el origen, entonces use el código deserializador (bloque 2). Si Kafka es el receptor, use el código serializador (bloque 3).

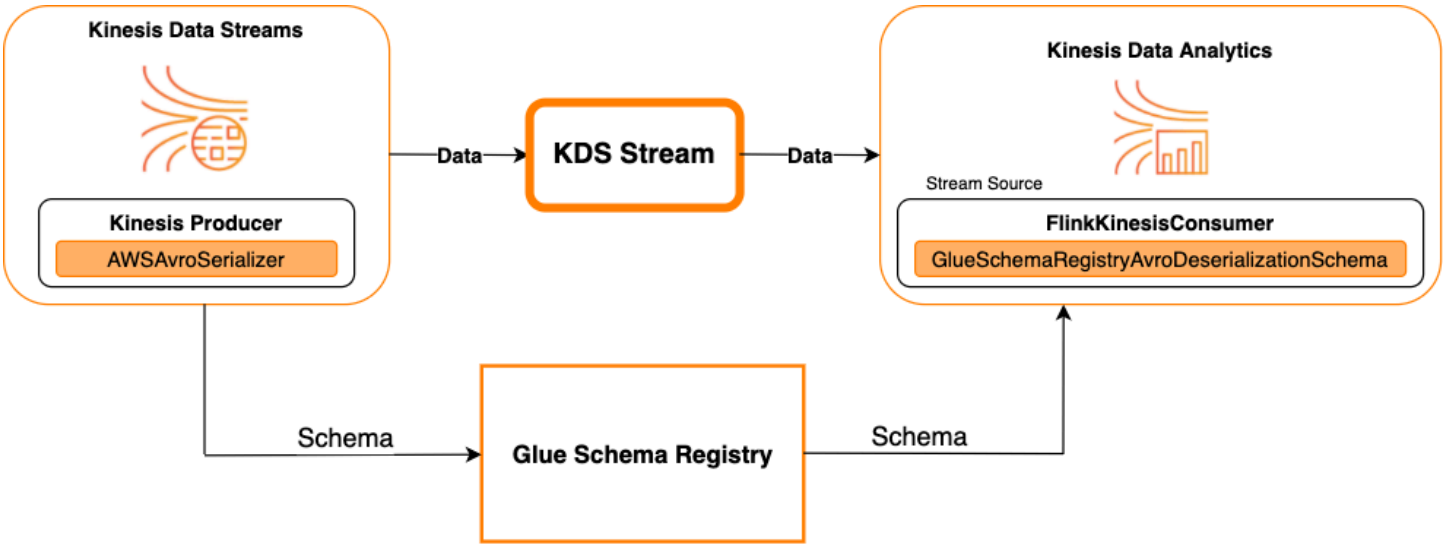
```
StreamExecutionEnvironment env = StreamExecutionEnvironment.getExecutionEnvironment();

String topic = "topic";
Properties properties = new Properties();
properties.setProperty("bootstrap.servers", "localhost:9092");
properties.setProperty("group.id", "test");

// block 1
Map<String, Object> configs = new HashMap<>();
configs.put(AWSSchemaRegistryConstants.AWS_REGION, "aws-region");
configs.put(AWSSchemaRegistryConstants.SCHEMA_AUTO_REGISTRATION_SETTING, true);
configs.put(AWSSchemaRegistryConstants.AVRO_RECORD_TYPE,
 AvroRecordType.GENERIC_RECORD.getName());

FlinkKafkaConsumer<GenericRecord> consumer = new FlinkKafkaConsumer<>(
    topic,
    // block 2
    GlueSchemaRegistryAvroDeserializationSchema.forGeneric(schema, configs),
    properties);

FlinkKafkaProducer<GenericRecord> producer = new FlinkKafkaProducer<>(
    topic,
    // block 3
    GlueSchemaRegistryAvroSerializationSchema.forGeneric(schema, topic, configs),
    properties);

DataStream<GenericRecord> stream = env.addSource(consumer);
stream.addSink(producer);
env.execute();
```

Integración de Kinesis Data Streams con Apache Flink

Puede usar Managed Service para Apache Flink, con Kinesis Data Streams como origen o como receptor.

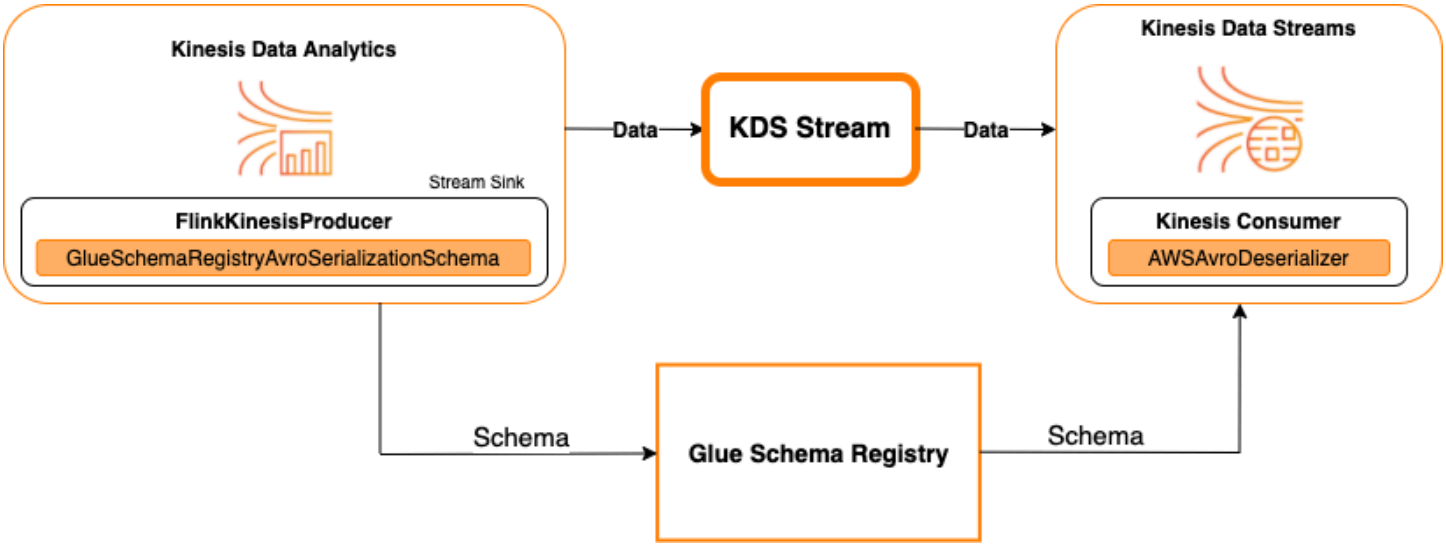
Kinesis Data Streams como origen

En el siguiente diagrama, se muestra la integración de Kinesis Data Streams con Managed Service para Apache Flink, con Kinesis Data Streams como origen.



Kinesis Data Streams como receptor

En el siguiente diagrama, se muestra la integración de Kinesis Data Streams con Managed Service para Apache Flink, con Kinesis Data Streams como receptor.



Para integrar Kinesis Data Streams con Managed Service para Apache Flink, con Kinesis Data Streams como origen o receptor, realice los cambios de código que se indican a continuación. Agregue los bloques de código en negrita a su código respectivo en las secciones análogas.

Si Kinesis Data Streams es el origen, utilice el código deserializador (bloque 2). Si Kinesis Data Streams es el receptor, use el código serializador (bloque 3).

```
StreamExecutionEnvironment env = StreamExecutionEnvironment.getExecutionEnvironment();

String streamName = "stream";
Properties consumerConfig = new Properties();
consumerConfig.put(AWSConfigConstants.AWS_REGION, "aws-region");
consumerConfig.put(AWSConfigConstants.AWS_ACCESS_KEY_ID, "aws_access_key_id");
consumerConfig.put(AWSConfigConstants.AWS_SECRET_ACCESS_KEY, "aws_secret_access_key");
consumerConfig.put(ConsumerConfigConstants.STREAM_INITIAL_POSITION, "LATEST");

// block 1
Map<String, Object> configs = new HashMap<>();
configs.put(AWSSchemaRegistryConstants.AWS_REGION, "aws-region");
configs.put(AWSSchemaRegistryConstants.SCHEMA_AUTO_REGISTRATION_SETTING, true);
configs.put(AWSSchemaRegistryConstants.AVRO_RECORD_TYPE,
 AvroRecordType.GENERIC_RECORD.getName());

FlinkKinesisConsumer<GenericRecord> consumer = new FlinkKinesisConsumer<>(
    streamName,
    // block 2
    GlueSchemaRegistryAvroDeserializationSchema.forGeneric(schema, configs),
    properties);

FlinkKinesisProducer<GenericRecord> producer = new FlinkKinesisProducer<>(
    // block 3
    GlueSchemaRegistryAvroSerializationSchema.forGeneric(schema, topic, configs),
    properties);
producer.setDefaultStream(streamName);
producer.setDefaultPartition("0");

DataStream<GenericRecord> stream = env.addSource(consumer);
stream.addSink(producer);
env.execute();
```

Caso de uso: integración con AWS Lambda

Para utilizar una función AWS Lambda como consumidor Apache Kafka/Amazon MSK y deserializar mensajes codificados por AVRO-con AWS Glue Schema Registry, visite la página de MSK Labs.

Caso de uso: AWS Glue Data Catalog

Las tablas de AWS Glue soportan esquemas que se pueden especificar en forma manual o por referencia a AWS Glue Schema Registry. Schema Registry se integra con el Catálogo de datos para

permitirle utilizar opcionalmente esquemas almacenados en Schema Registry al crear o actualizar tablas o particiones de AWS Glue en el Catálogo de datos. Para identificar una definición de esquema en Schema Registry, es necesario conocer, al menos, el ARN del esquema del que forma parte. Una versión de esquema, que contiene una definición de esquema, puede ser referenciada por su UUID o número de versión. Siempre hay una versión de esquema, la “última” versión, que se puede buscar sin saber su número de versión o UUID.

Al llamar a las operaciones `CreateTable` o `UpdateTable`, transferirá una estructura `TableInput` que contiene un `StorageDescriptor`, que podría tener una `SchemaReference` a un esquema existente en Schema Registry. Del mismo modo, cuando se llama a las API `GetTable` o `GetPartition`, la respuesta puede contener el esquema y la `SchemaReference`. Cuando se crea una tabla o partición mediante referencias de esquema, el Catálogo de datos intentará buscar el esquema para esta referencia de esquema. En caso de que no pueda encontrar el esquema, Schema Registry devuelve un esquema vacío en la respuesta `GetTable`; de lo contrario, la respuesta tendrá el esquema y la referencia del esquema.

Puede realizar las siguientes acciones desde la consola de AWS Glue.

Para realizar estas operaciones y crear, actualizar o ver la información del esquema, debe brindar al usuario que realiza la llamada un rol de IAM que proporcione permisos para la API `GetSchemaVersion`.

Agregar una tabla o actualizar el esquema de una tabla

Agregar una nueva tabla a partir de un esquema existente enlaza la tabla a una versión de esquema específica. Una vez que se registren las nuevas versiones de esquema, puede actualizar esta definición de tabla desde la página View tables (Ver tabla) en la consola de AWS Glue o con la API UpdateTable acción (Python: update_table).

Agregar una tabla a partir de un esquema existente

Puede crear una tabla de AWS Glue a partir de una versión de esquema en el registro mediante la consola AWS Glue o la API `CreateTable`.

API de AWS Glue

Al llamar a la API `CreateTable`, transferirá una `TableInput` que contiene un `StorageDescriptor` con una `SchemaReference` a un esquema existente en Schema Registry.

Consola de AWS Glue

Para crear una tabla desde la consola de AWS Glue:

1. Inicie sesión en AWS Management Console y abra la consola de AWS Glue en https://console.aws.amazon.com/glue/.
2. En el panel de navegación, en Data catalog (Catálogo de datos), elija Tables (Tablas).
3. En el menú Add tables (Agregar tablas), elija Add table from existing schema (Agregar tabla a partir del esquema existente).
4. Configure las propiedades de la tabla y el almacén de datos según la Guía para desarrolladores de AWS Glue.
5. En la página Choose a Glue schema (Elegir un esquema de Glue), seleccione el Record (Registro) donde reside el esquema.
6. Elija el Schema name (Nombre del esquema) y seleccione la Version (Versión) del esquema que se va a aplicar.
7. Revise la previsualización del esquema y elija Next (Siguiente).
8. Revise y cree la tabla.

El esquema y la versión aplicados a la tabla aparecen en la columna Glue schema (Esquema de Glue) en la lista de tablas. Puede ver la tabla para ver más detalles.

Actualización del esquema de una tabla

Cuando esté disponible una nueva versión de esquema, es posible que desee actualizar el esquema de una tabla mediante la API UpdateTable acción (Python: update_table) o la consola de AWS Glue.

> **Important**
>
> Al actualizar el esquema de una tabla existente que tiene un esquema de AWS Glue especificado manualmente, el nuevo esquema al que se hace referencia en el Schema Registry puede ser incompatible. Esto puede dar lugar a que sus trabajos presenten errores.

API de AWS Glue

Al llamar a la API `UpdateTable`, transferirá una `TableInput` que contiene un `StorageDescriptor` con una `SchemaReference` a un esquema existente en Schema Registry.

Consola de AWS Glue

Para actualizar el esquema de una tabla desde la consola de AWS Glue:

1. Inicie sesión en AWS Management Console y abra la consola de AWS Glue en https://console.aws.amazon.com/glue/.
2. En el panel de navegación, en Data catalog (Catálogo de datos), elija Tables (Tablas).
3. Vea la tabla de la lista de tablas.
4. Haga clic en Update schema (Actualizar esquema) en el cuadro que le informa sobre una nueva versión.
5. Revise las diferencias entre el esquema actual y el nuevo.
6. Seleccione Show all schema differences (Mostrar todas las diferencias de esquemas) para ver más detalles.
7. Seleccione Save table (Guardar tabla) para aceptar la nueva versión.

Caso de uso: Streaming de AWS Glue

El streaming de AWS Glue consume datos de orígenes de streaming y realiza operaciones ETL antes de escribir en un receptor de salida. El origen del streaming de entrada se puede especificar mediante una tabla de datos o directamente especificando la configuración de origen.

El streaming de AWS Glue admite una tabla del Catálogo de datos para el origen de transmisión creado con el esquema presente en AWS Glue Schema Registry. Puede crear un esquema en AWS Glue Schema Registry y, mediante el uso de este, crear una tabla de AWS Glue con un origen de streaming. Esta tabla de AWS Glue se puede utilizar como entrada para un trabajo de streaming de AWS Glue de manera de deserializar los datos en el flujo de entrada.

Un punto que se debe tener en cuenta aquí es que, cuando cambia el esquema de AWS Glue Schema Registry, debe reiniciar el trabajo de streaming de AWS Glue para que el cambio se vea reflejado.

Caso de uso: Apache Kafka Streams

La API Apache Kafka Streams es una biblioteca cliente para procesar y analizar datos almacenados en Apache Kafka. Esta sección describe la integración de Apache Kafka Streams con AWS Glue Schema Registry, que le permite administrar y aplicar esquemas en sus aplicaciones de streaming de datos. Para obtener más información sobre Apache Kafka Streams, consulte Apache Kafka Streams.

Integración con las bibliotecas SerDes

Existe una clase de `GlueSchemaRegistryKafkaStreamsSerde` con la que puede configurar una aplicación de Streams.

Código de ejemplo de aplicación de Kafka Streams

Para utilizar el AWS Glue Schema Registry dentro de una aplicación Apache Kafka Streams:

1. Configure la aplicación Kafka Streams.

```
final Properties props = new Properties();
    props.put(StreamsConfig.APPLICATION_ID_CONFIG, "avro-streams");
    props.put(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9092");
    props.put(StreamsConfig.CACHE_MAX_BYTES_BUFFERING_CONFIG, 0);
    props.put(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG,
 Serdes.String().getClass().getName());
    props.put(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG,
 AWSKafkaAvroSerDe.class.getName());
    props.put(ConsumerConfig.AUTO_OFFSET_RESET_CONFIG, "earliest");

    props.put(AWSSchemaRegistryConstants.AWS_REGION, "aws-region");
    props.put(AWSSchemaRegistryConstants.SCHEMA_AUTO_REGISTRATION_SETTING, true);
    props.put(AWSSchemaRegistryConstants.AVRO_RECORD_TYPE,
 AvroRecordType.GENERIC_RECORD.getName());
 props.put(AWSSchemaRegistryConstants.DATA_FORMAT, DataFormat.AVRO.name());
```

2. Cree un flujo a partir del tema avro-input.

```
StreamsBuilder builder = new StreamsBuilder();
final KStream<String, GenericRecord> source = builder.stream("avro-input");
```

3. Procese los registros de datos (el ejemplo filtra aquellos registros cuyo valor de color_favorito es rosa o cuyo valor de cantidad es 15).

```
final KStream<String, GenericRecord> result = source
    .filter((key, value) -
> !"pink".equals(String.valueOf(value.get("favorite_color"))));
    .filter((key, value) -> !"15.0".equals(String.valueOf(value.get("amount"))));
```

4. Escriba los resultados en el tema avro-output.

```
result.to("avro-output");
```

5. Inicie la aplicación Apache Kafka Streams.

```
KafkaStreams streams = new KafkaStreams(builder.build(), props);
streams.start();
```

Resultados de implementación

Estos resultados muestran el proceso de filtrado de registros que se filtraron en el paso 3 como color_favorito “rosa” o valor “15,0”.

Registros antes del filtrado:

```
{"name": "Sansa", "favorite_number": 99, "favorite_color": "white"}
{"name": "Harry", "favorite_number": 10, "favorite_color": "black"}
{"name": "Hermione", "favorite_number": 1, "favorite_color": "red"}
{"name": "Ron", "favorite_number": 0, "favorite_color": "pink"}
{"name": "Jay", "favorite_number": 0, "favorite_color": "pink"}

{"id": "commute_1","amount": 3.5}
{"id": "grocery_1","amount": 25.5}
{"id": "entertainment_1","amount": 19.2}
{"id": "entertainment_2","amount": 105}
 {"id": "commute_1","amount": 15}
```

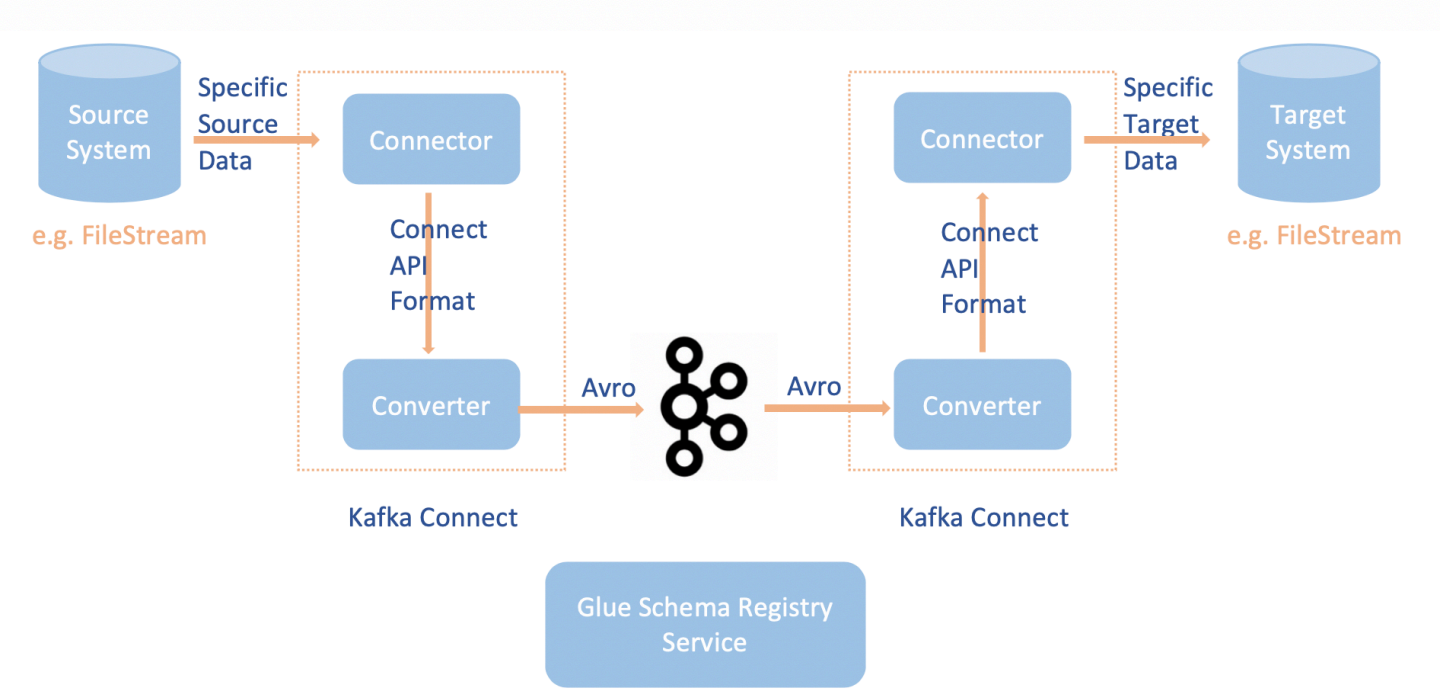
Registros después del filtrado:

```
{"name": "Sansa", "favorite_number": 99, "favorite_color": "white"}
{"name": "Harry", "favorite_number": 10, "favorite_color": "black"}
{"name": "Hermione", "favorite_number": 1, "favorite_color": "red"}
{"name": "Ron", "favorite_number": 0, "favorite_color": "pink"}

{"id": "commute_1","amount": 3.5}
{"id": "grocery_1","amount": 25.5}
{"id": "entertainment_1","amount": 19.2}
```

```
{"id": "entertainment_2","amount": 105}
```

Caso de uso: Apache Kafka Connect

La integración de Apache Kafka Connect con el AWS Glue Schema Registry permite obtener información de esquemas a partir de los conectores. Los convertidores Apache Kafka especifican el formato de datos dentro de Apache Kafka y cómo traducirlos a datos Apache Kafka Connect. Cada usuario de Apache Kafka Connect tendrá que configurar estos convertidores en función del formato en el que desea que sus datos estén cargados o almacenados en Apache Kafka. De esta manera, puede definir sus propios convertidores para traducir los datos de Apache Kafka Connect al tipo utilizado en AWS Glue Schema Registry (por ejemplo: Avro) y utilizar nuestro serializador para registrar su esquema y serializar. Los convertidores también pueden usar nuestro deserializador para deserializar los datos recibidos de Apache Kafka y volver a convertirlos en datos Apache Kafka Connect. A continuación se muestra un diagrama de flujo de trabajo de ejemplo.



1. Instale el proyecto `aws-glue-schema-registry` al clonar el [repositorio de Github para AWS Glue Schema Registry](#).

```
git clone git@github.com:awslabs/aws-glue-schema-registry.git
cd aws-glue-schema-registry
mvn clean install
```

```
mvn dependency:copy-dependencies
```

2. Si planea usar Apache Kafka Connect en modo independiente, actualice connect-standalone.properties según las instrucciones que se incluyen a continuación. Si planea usar Apache Kafka Connect en modo distribuido, actualice connect-avro-distributed.properties según las mismas instrucciones.

   a. Agregue estas propiedades también al archivo de propiedades de conexión Apache Kafka:

   ```
   key.converter.region=aws-region
   value.converter.region=aws-region
   key.converter.schemaAutoRegistrationEnabled=true
   value.converter.schemaAutoRegistrationEnabled=true
   key.converter.avroRecordType=GENERIC_RECORD
   value.converter.avroRecordType=GENERIC_RECORD
   ```

   b. Agregue el siguiente comando a la sección Launch mode (Modo de lanzamiento) en kafka-run-class.sh:

   ```
   -cp $CLASSPATH:"<your AWS GlueSchema Registry base directory>/target/dependency/*"
   ```

3. Agregue el siguiente comando a la sección Launch mode (Modo de lanzamiento) en kafka-run-class.sh

   ```
   -cp $CLASSPATH:"<your AWS GlueSchema Registry base directory>/target/dependency/*"
   ```

   Debería tener un aspecto similar al siguiente:

   ```
   # Launch mode
   if [ "x$DAEMON_MODE" = "xtrue" ]; then
     nohup "$JAVA" $KAFKA_HEAP_OPTS $KAFKA_JVM_PERFORMANCE_OPTS $KAFKA_GC_LOG_OPTS
    $KAFKA_JMX_OPTS $KAFKA_LOG4J_OPTS -cp $CLASSPATH:"/Users/johndoe/aws-glue-schema-
   registry/target/dependency/*" $KAFKA_OPTS "$@" > "$CONSOLE_OUTPUT_FILE" 2>&1 < /dev/
   null &
   else
     exec "$JAVA" $KAFKA_HEAP_OPTS $KAFKA_JVM_PERFORMANCE_OPTS $KAFKA_GC_LOG_OPTS
    $KAFKA_JMX_OPTS $KAFKA_LOG4J_OPTS -cp $CLASSPATH:"/Users/johndoe/aws-glue-schema-
   registry/target/dependency/*" $KAFKA_OPTS "$@"
   fi
   ```

4. Si usa bash, ejecute los siguientes comandos para configurar su CLASSPATH en su bash_profile. Para cualquier otro shell, actualice el entorno en consecuencia.

```
echo 'export GSR_LIB_BASE_DIR=<>' >>~/.bash_profile
echo 'export GSR_LIB_VERSION=1.0.0' >>~/.bash_profile
echo 'export KAFKA_HOME=<your Apache Kafka installation directory>' >>~/.bash_profile
echo 'export CLASSPATH=$CLASSPATH:$GSR_LIB_BASE_DIR/avro-kafkaconnect-converter/
target/schema-registry-kafkaconnect-converter-$GSR_LIB_VERSION.jar:$GSR_LIB_BASE_DIR/
common/target/schema-registry-common-$GSR_LIB_VERSION.jar:$GSR_LIB_BASE_DIR/
avro-serializer-deserializer/target/schema-registry-serde-$GSR_LIB_VERSION.jar'
 >>~/.bash_profile
source ~/.bash_profile
```

5. (Opcional) si desea realizar una prueba con un origen de archivo simple, clone el conector de origen del archivo.

```
git clone https://github.com/mmolimar/kafka-connect-fs.git
cd kafka-connect-fs/
```

   a. En la configuración del conector de origen, edite el formato de datos a Avro, el lector de archivos a `AvroFileReader` y actualice un objeto Avro de ejemplo desde la ruta del archivo de la que está leyendo. Por ejemplo:

```
vim config/kafka-connect-fs.properties
```

```
fs.uris=<path to a sample avro object>
policy.regexp=^.*\.avro$
file_reader.class=com.github.mmolimar.kafka.connect.fs.file.reader.AvroFileReader
```

   b. Instale el conector de origen.

```
mvn clean package
echo "export CLASSPATH=\$CLASSPATH:\"\$(find target/ -type f -name '*.jar'| grep
 '\-package' | tr '\n' ':')\"" >>~/.bash_profile
source ~/.bash_profile
```

   c. Actualice las propiedades del receptor en *<your Apache Kafka installation directory>*`/config/connect-file-sink.properties`, actualice el nombre del tema y el nombre del archivo de salida.

```
file=<output file full path>
topics=<my topic>
```

6. Inicie el conector de origen (en este ejemplo es un conector de origen de archivo).

```
$KAFKA_HOME/bin/connect-standalone.sh $KAFKA_HOME/config/connect-
standalone.properties config/kafka-connect-fs.properties
```

7. Ejecute el conector del receptor (en este ejemplo es un conector receptor de archivo).

```
$KAFKA_HOME/bin/connect-standalone.sh $KAFKA_HOME/config/connect-
standalone.properties $KAFKA_HOME/config/connect-file-sink.properties
```

Para ver un ejemplo de uso de Kafka Connect, mire el script run-local-tests.sh en la carpeta integration-tests (pruebas de integración) en el repositorio de Github para AWS Glue Schema Registry.

Migración desde un registro de esquemas de terceros a AWS Glue Schema Registry

La migración de un registro de esquema de terceros a AWS Glue Schema Registry tiene una dependencia en el registro de esquemas de terceros existente. Si hay registros en un tema de Apache Kafka que se enviaron mediante un registro de esquemas de terceros, los consumidores necesitan el registro de esquemas de terceros para deserializar esos registros. `AWSKafkaAvroDeserializer` proporciona la capacidad de especificar una clase de deserializador secundario que apunta al deserializador de terceros y se utiliza para deserializar esos registros.

Existen dos criterios para retirar un esquema de terceros. En primer lugar, el retiro sólo puede ocurrir después de que los registros en temas de Apache Kafka que utilizan el registro de esquemas de terceros ya no sean requeridos por o para un consumidor. En segundo lugar, el retiro puede ocurrir si se eliminan los temas de Apache Kafka, dependiendo del período de retención especificado para esos temas. Tenga en cuenta que si tiene temas que tienen retención infinita, puede migrar a AWS Glue Schema Registry, pero no podrá retirar el registro de esquemas de terceros. Como solución alternativa, puede usar una aplicación o Mirror Maker 2 para leer el tema actual y producir un tema nuevo con AWS Glue Schema Registry.

Migración desde un registro de esquemas de terceros a AWS Glue Schema Registry:

1. Cree un registro en AWS Glue Schema Registry o utilice el registro predeterminado.
2. Detenga el consumidor. Modifíquelo para incluir AWS Glue Schema Registry como el deserializador principal y el registro de esquemas de terceros como el secundario.
   - Configure las propiedades del consumidor. En este ejemplo, el secondary_deserializer (deserializador secundario) se configura en un deserializador diferente. El comportamiento es el siguiente: el consumidor recupera los registros de Amazon MSK y primero intenta utilizar el `AWSKafkaAvroDeserializer`. Si no puede leer el byte mágico que contiene el ID de esquema de Avro para AWS Glue Schema Registry, el `AWSKafkaAvroDeserializer` luego intenta usar la clase de deserializador proporcionada en el secondary_deserializer (deserializador secundario). Las propiedades específicas del deserializador secundario también deben proporcionarse en las propiedades del consumidor, como schema_registry_url_config (configuración url de registro de esquema) y specific_avro_reader_config (configuración de lector avro específica), como se muestra a continuación.

```
consumerProps.setProperty(ConsumerConfig.KEY_DESERIALIZER_CLASS_CONFIG,
 StringDeserializer.class.getName());
consumerProps.setProperty(ConsumerConfig.VALUE_DESERIALIZER_CLASS_CONFIG,
 AWSKafkaAvroDeserializer.class.getName());
consumerProps.setProperty(AWSSchemaRegistryConstants.AWS_REGION,
 KafkaClickstreamConsumer.gsrRegion);
consumerProps.setProperty(AWSSchemaRegistryConstants.SECONDARY_DESERIALIZER,
 KafkaAvroDeserializer.class.getName());
consumerProps.setProperty(KafkaAvroDeserializerConfig.SCHEMA_REGISTRY_URL_CONFIG,
 "URL for third-party schema registry");
consumerProps.setProperty(KafkaAvroDeserializerConfig.SPECIFIC_AVRO_READER_CONFIG,
 "true");
```

3. Reinicie el consumidor.
4. Detenga el productor y dirija al productor a AWS Glue Schema Registry.
   a. Configure las propiedades del productor. En este ejemplo, el productor utilizará las versiones de esquema de registro predeterminado y registro automático.

```
producerProps.setProperty(ProducerConfig.KEY_SERIALIZER_CLASS_CONFIG,
 StringSerializer.class.getName());
producerProps.setProperty(ProducerConfig.VALUE_SERIALIZER_CLASS_CONFIG,
 AWSKafkaAvroSerializer.class.getName());
producerProps.setProperty(AWSSchemaRegistryConstants.AWS_REGION, "us-east-2");
```

```
producerProps.setProperty(AWSSchemaRegistryConstants.AVRO_RECORD_TYPE,
 AvroRecordType.SPECIFIC_RECORD.getName());
producerProps.setProperty(AWSSchemaRegistryConstants.SCHEMA_AUTO_REGISTRATION_SETTING,
 "true");
```

5. (Opcional) mueva manualmente los esquemas y las versiones de esquema existentes del registro de esquemas de terceros actual a AWS Glue Schema Registry, ya sea al registro predeterminado en AWS Glue Schema Registry o a un registro no predeterminado específico en AWS Glue Schema Registry. Esto se puede hacer al exportar esquemas de los registros de esquemas de terceros en formato JSON y crear nuevos esquemas en AWS Glue Schema Registry con AWS Management Console o AWS CLI.

   Este paso puede ser importante si necesita habilitar las comprobaciones de compatibilidad con versiones de esquema anteriores para las versiones de esquema recién creadas mediante AWS CLI y AWS Management Console, o cuando los productores envían mensajes con un nuevo esquema con la función de registro automático de versiones de esquema activada.
6. Inicie el productor.

Conexión a datos

Una AWS Glue conexión es un objeto del catálogo de datos que almacena las credenciales de inicio de sesión, las cadenas de URI, la información de la nube privada virtual (VPC) y más para un almacén de datos concreto. AWS Glue los rastreadores, los trabajos y los puntos finales de desarrollo utilizan conexiones para acceder a determinados tipos de almacenes de datos. Puede utilizar conexiones para orígenes y destinos, así como reutilizar la misma conexión en varios trabajos de rastreador o de extracción, transformación y carga (ETL).

AWS Glue admite los siguientes tipos de conexión:

- Amazon DocumentDB
- Amazon OpenSearch Service, para usar con AWS Glue Spark.
- Amazon Redshift
- Azure Cosmos, para usar Azure Cosmos DB para NoSQL con AWS Glue trabajos de ETL
- Azure SQL, para usar con AWS Glue Spark.
- Google BigQuery, para usar con AWS Glue Spark.
- JDBC
- Kafka
- MongoDB
- MongoDB Atlas
- Salesforce
- SAP HANA, para usar con AWS Glue Spark.
- Snowflake, para usar con AWS Glue Spark.
- Teradata Vantage, cuando se usa para Spark. AWS Glue
- Vertica, para usar con Spark. AWS Glue
- Varias ofertas de Amazon Relational Database Service (Amazon RDS).
- Network (Red) (designa una conexión a un origen de datos que se encuentra en un entorno de Amazon Virtual Private Cloud [Amazon VPC])
- Aurora (compatible si se utiliza el controlador JDBC nativo. No se pueden aprovechar todas las características del controlador)

Con AWS Glue Studio, también se puede crear una conexión para un conector. Un conector es un paquete de códigos opcionales que ayuda a acceder a almacenes de datos en AWS Glue Studio. Para obtener más información, consulte Uso de conectores y conexiones con AWS Glue Studio

Para obtener información sobre cómo conectarse a bases de datos locales, consulte Cómo acceder a los almacenes de datos locales y analizarlos mediante el sitio web del AWS Glue blog sobre AWS macrodatos.

En esta sección se incluyen los siguientes temas para ayudarle a utilizar las conexiones de AWS Glue :

- Propiedades de las conexiones de AWS Glue
- Almacenamiento de credenciales de conexión en AWS Secrets Manager
- Adición de una conexión de AWS Glue
- Prueba de una conexión de AWS Glue
- Configuración de las llamadas de AWS para que pasen a través de la VPC
- Conexión a un almacén de datos de JDBC en una VPC
- Uso de una conexión MongoDB o MongoDB Atlas
- Rastreo de un almacén de datos de Amazon S3 mediante un punto de conexión de VPC
- Solución de problemas de conexión en AWS Glue
- Tutorial: uso de AWS Glue Connector for Elasticsearch

Propiedades de las conexiones de AWS Glue

En este tema se incluye información sobre las propiedades de AWS Glue las conexiones.

Temas

- Propiedades de conexión requeridas
- Propiedades de las conexiones JDBC de AWS Glue
- Propiedades de conexión de MongoDB y MongoDB Atlas de AWS Glue
- Propiedades de conexión de Salesforce
- Conexión Snowflake
- Conexión vertical
- Conexión a SAP HANA

- Conexión a Azure SQL
- Conexión de Teradata Vantage
- OpenSearch Conexión de servicio
- Conexión de Azure Cosmos
- Propiedades de las conexiones SSL de AWS Glue
- Propiedades de las conexiones de Apache Kafka para la autenticación de clientes
- BigQuery Conexión a Google
- Conexión vertical

Propiedades de conexión requeridas

Cuando se define una conexión en la consola de AWS Glue, se deben proporcionar valores para las siguientes propiedades:

Nombre de la conexión

Ingrese un nombre único para la conexión.

Tipo de conexión

Elija JDBC o uno de los tipos de conexión específicos.

Para obtener más información sobre el tipo de conexión JDBC, consulte the section called “Propiedades de las conexiones JDBC”

Elija Red para conectarse a un origen de datos dentro de un entorno de Amazon Virtual Private Cloud (Amazon VPC).

En función del tipo que elija, la consola de AWS Glue muestra otros campos obligatorios. Por ejemplo, si elige Amazon RDS, debe elegir el motor de base de datos.

Exigir conexión SSL

Cuando selecciona esta opción, AWS Glue debe comprobar que se realizó la conexión a la base de datos a través de una conexión de capa de conexión segura (SSL) de confianza.

Para obtener más información, incluidas las opciones adicionales que están disponibles al seleccionar esta opción, consulte the section called “Propiedades de las conexiones SSL”.

Seleccione el clúster MSK [solo streaming administrada por Amazon para Apache Kafka (MSK)]

Especifica un clúster de MSK de otra AWS cuenta.

URL de servidor de arranque Kafka (sólo Kafka)

Especifica una lista separada por comas de direcciones URL del servidor de arranque. Incluya el número de puerto. Por ejemplo: b-1.vpc-test-2.o4q88o.c6.kafka.us-east-1.amazonaws.com:9094, b-2.vpc-test-2.o4q88o.c6.kafka.us-east-1.amazonaws.com:9094, b-3.vpc-test-2.o4q88o.c6.kafka.us-east-1.amazonaws.com:9094

Propiedades de las conexiones JDBC de AWS Glue

AWS Glue puede conectarse a los siguientes almacenes de datos a través de una conexión JDBC:

- Amazon Redshift
- Amazon Aurora
- Microsoft SQL Server
- MySQL
- Oracle
- PostgreSQL
- Snowflake, cuando se utilizan rastreadores. AWS Glue
- Aurora (compatible si se utiliza el controlador JDBC nativo. No se pueden aprovechar todas las características del controlador)
- Amazon RDS for MariaDB

Important

Actualmente, los trabajos de ETL pueden utilizar conexiones JDBC dentro de una sola subred. Si tiene varios almacenes de datos en un trabajo, deben estar en la misma subred, o ser accesibles desde la subred.
Si decide incorporar sus versiones de controladores JDBC para rastreadores AWS Glue, estos consumirán recursos en trabajos AWS Glue y Amazon S3 para garantizar que los controladores proporcionados se ejecuten en su entorno. El uso adicional de los recursos se reflejará en su cuenta. Además, proporcionar su propio controlador JDBC no significa que

el rastreador pueda aprovechar todas las funciones del controlador. Los controladores se limitan a las propiedades descritas en Definir las conexiones en el catálogo de datos.

A continuación, se muestran propiedades adicionales para el tipo de conexión JDBC.

JDBC URL (Dirección URL de JDBC)

Ingrese la dirección URL para el almacén de datos de JDBC. Para la mayoría de motores de base de datos, este campo se encuentra en el siguiente formato. En este formato, sustituya *protocolo*, *host*, *puerto* y *db_name* con su propia información.

`jdbc:`*`protocol`*`://`*`host`*`:`*`port`*`/`*`db_name`*

En función del motor de base de datos, es posible que se requiera un formato de dirección URL de JDBC diferente. Este formato puede utilizar los dos puntos (:) y la barra inclinada (/) de forma ligeramente diferente o palabras clave distintas para especificar bases de datos.

Para que JDBC se conecte al almacén de datos, se requiere un `db_name` en el almacén de datos. Se utiliza `db_name` para establecer una conexión de red con los ajustes `username` y `password` proporcionados. Cuando se haya establecido la conexión, AWS Glue tendrá acceso a otras bases de datos en el almacén de datos para ejecutar un rastreador o un flujo de trabajo de ETL.

Los siguientes ejemplos de dirección URL de JDBC muestran la sintaxis para diversos motores de base de datos.

- Para conectarse a un almacén de datos de clústeres de Amazon Redshift con una base de datos de `dev`:

  ```
  jdbc:redshift://xxx.us-east-1.redshift.amazonaws.com:8192/dev
  ```

- Para conectarse a un almacén de datos de Amazon RDS for MySQL con una base de datos de `employee`:

  ```
  jdbc:mysql://xxx-cluster.cluster-xxx.us-east-1.rds.amazonaws.com:3306/employee
  ```

- Para conectarse a un almacén de datos de Amazon RDS for PostgreSQL con una base de datos de `employee`:

  ```
  jdbc:postgresql://xxx-cluster.cluster-xxx.us-east-1.rds.amazonaws.com:5432/employee
  ```

- Para conectarse a un almacén de datos de Amazon RDS for Oracle con un nombre de servicio `employee`:

  ```
  jdbc:oracle:thin://@xxx-cluster.cluster-xxx.us-
  east-1.rds.amazonaws.com:1521/employee
  ```

  La sintaxis para Amazon RDS for Oracle puede seguir los siguientes patrones. En estos patrones, sustituya *`host`*, *`puerto`*, *`nombre de servicio`* y *`SID`* con su propia información.

  - `jdbc:oracle:thin://@`*`host`*`:`*`port`*`/`*`service_name`*
  - `jdbc:oracle:thin://@`*`host`*`:`*`port`*`:`*`SID`*

- Para conectarse a un almacén de datos de Amazon RDS for Microsoft SQL Server con una base de datos de `employee`:

  ```
  jdbc:sqlserver://xxx-cluster.cluster-xxx.us-
  east-1.rds.amazonaws.com:1433;databaseName=employee
  ```

  La sintaxis para Amazon RDS for SQL Server puede seguir los siguientes patrones. En estos patrones, reemplace *`server_name`*, *`port`* y *`db_name`* por su propia información.

  - `jdbc:sqlserver://`*`server_name`*`:`*`port`*`;database=`*`db_name`*
  - `jdbc:sqlserver://`*`server_name`*`:`*`port`*`;databaseName=`*`db_name`*

- Para conectarse a una Amazon Aurora PostgreSQL instancia de la `employee` base de datos, especifique el punto final de la instancia de base de datos, el puerto y el nombre de la base de datos:

  ```
  jdbc:postgresql://employee_instance_1.xxxxxxxxxxxx.us-
  east-2.rds.amazonaws.com:5432/employee
  ```

- Para conectarse a un banco de Amazon RDS for MariaDB datos con una `employee` base de datos, especifique el punto final de la instancia de base de datos, el puerto y el nombre de la base de datos:

  ```
  jdbc:mysql://xxx-cluster.cluster-xxx.aws-
  region.rds.amazonaws.com:3306/employee
  ```

- > **Warning**
  >
  > Las conexiones JDBC de Snowflake solo las admiten los rastreadores. AWS Glue Cuando utilice el conector Snowflake en los AWS Glue trabajos, utilice el tipo de conexión Snowflake.

  Para conectarse a una instancia de Snowflake de la base de datos `sample`, especifique el punto de conexión para la instancia de Snowflake, el usuario, el nombre de la base de datos y el nombre del rol. Si lo desea, puede agregar el parámetro `warehouse`.

  ```
  jdbc:snowflake://account_name.snowflakecomputing.com/?
  user=user_name&db=sample&role=role_name&warehouse=warehouse_name
  ```

  > **Important**
  >
  > Para las conexiones de Snowflake a través de JDBC, se aplica el orden de los parámetros de la URL y se deben ordenar como `user`, db, `role_name` y `warehouse`.

- Para conectarse a una instancia de Snowflake de la `sample` base de datos con un enlace AWS privado, especifique la URL JDBC de Snowflake de la siguiente manera:

  ```
  jdbc:snowflake://account_name.region.privatelink.snowflakecomputing.com/?
  user=user_name&db=sample&role=role_name&warehouse=warehouse_name
  ```

Nombre de usuario

> **Note**
>
> Le recomendamos que utilice un AWS secreto para almacenar las credenciales de conexión en lugar de proporcionar su nombre de usuario y contraseña directamente. Para obtener más información, consulte Almacenamiento de credenciales de conexión en AWS Secrets Manager.

Proporcione un nombre de usuario que tenga permisos para obtener acceso al almacén de datos de JDBC.

Password

Ingrese la contraseña para el nombre de usuario con los permisos de acceso al almacén de datos de JDBC.

Puerto

Ingrese el puerto que se utiliza en la URL de JDBC para conectarse a una instancia de Oracle de Amazon RDS. Este campo es solo se muestra cuando se selecciona Solicitar conexión SSL para una instancia de Oracle de Amazon RDS.

VPC

Seleccione el nombre de la nube privada virtual (VPC) que contenga el almacén de datos. La consola de AWS Glue muestra todas las VPC para la región actual.

> **Important**
>
> Al trabajar con una conexión JDBC alojada fuera de ella, por ejemplo AWS, con datos de Snowflake, la VPC debe tener una puerta de enlace NAT que divida el tráfico en subredes públicas y privadas. La subred pública se utiliza para la conexión a la fuente externa y la subred interna se utiliza para el procesamiento por parte de. AWS Glue Para obtener información sobre cómo configurar su Amazon VPC para conexiones externas, lea Conectarse a Internet u otras redes mediante dispositivos NAT y Configuración de una VPC de Amazon para conexiones JDBC a los almacenes de datos de Amazon RDS de AWS Glue.

Subred

Seleccione la subred dentro de la VPC que contenga el almacén de datos. La consola de AWS Glue incluye todas las subredes para el almacén de datos en la VPC.

Grupos de seguridad

Elija los grupos de seguridad asociados a su almacén de datos. AWS Glue requiere uno o varios grupos de seguridad con una regla de origen de entrada que permite AWS Glue establezca la conexión. La consola de AWS Glue incluye todos los grupos de seguridad con acceso de entrada a su VPC. AWS Glue asocia estos grupos de seguridad con la interfaz de red elástica que se asocia a su subred de VPC.

Nombre de la clase de controlador JDBC: opcional

Proporcione el nombre de la clase de controlador JDBC personalizado:

- Postgres – org.postgresql.Driver
- MySQL – com.mysql.jdbc.Driver, com.mysql.cj.jdbc.Driver
- Redshift – com.amazon.redshift.jdbc.Driver, com.amazon.redshift.jdbc42.Driver
- Oracle: oracle.jdbc.driver. OracleDriver
- Servidor SQL: com.Microsoft.SqlServer.jdbc.sql ServerDriver

Ruta S3 del controlador JDBC: opcional

Proporcione la ubicación de Amazon S3 al controlador JDBC personalizado. Esta es una ruta absoluta a un archivo .jar. Si desea disponer de sus propios controladores JDBC para conectarse a los orígenes de datos de las bases de datos compatibles con su rastreador, puede especificar valores para los parámetros `customJdbcDriverS3Path` y `customJdbcDriverClassName`. Utilizar un controlador JDBC suministrado por un cliente se limita a lo requerido Propiedades de conexión requeridas.

Propiedades de conexión de MongoDB y MongoDB Atlas de AWS Glue

A continuación, se muestran propiedades adicionales para el tipo de conexión de MongoDB o MongoDB Atlas.

URL de MongoDB

Ingrese la URL de su almacén de datos de MongoDB o MongoDB Atlas:

- Para MongoDB: mongodb://host:port/database. El host puede ser un nombre de host, una dirección IP o un socket de dominio UNIX. Si la cadena de conexión no especifica ningún puerto, utiliza el puerto predeterminado de MongoDB, 27017.
- Para MongoDB Atlas: mongodb+srv://server.example.com/database. El host puede ser un nombre de host que corresponde a un registro SRV de DNS. El formato SRV no requiere ningún puerto y utilizará el puerto MongoDB predeterminado, 27017.

Nombre de usuario

> **Note**
>
> Se recomienda utilizar un AWS secreto para almacenar las credenciales de conexión en lugar de proporcionar directamente el nombre de usuario y la contraseña. Para obtener más información, consulte Almacenamiento de credenciales de conexión en AWS Secrets Manager.

Proporcione un nombre de usuario que tenga permisos para obtener acceso al almacén de datos de JDBC.

Password

Ingrese la contraseña para el nombre de usuario con los permisos de acceso al almacén de datos de MongoDB o MongoDB Atlas.

Propiedades de conexión de Salesforce

Las siguientes son propiedades adicionales para el tipo de conexión de Salesforce.

- `ENTITY_NAME`(Cadena): (obligatorio) Se utiliza para lectura/escritura. El nombre de su objeto en Salesforce.
- `API_VERSION`(Cadena): (obligatorio) Se usa para lectura/escritura. Versión de la API Rest de Salesforce que desea utilizar.
- `SELECTED_FIELDS`(Lista<String>) - Predeterminada: vacía (SELECT *). Se usa para leer. Columnas que desee seleccionar para el objeto.
- `FILTER_PREDICATE`(Cadena) - Predeterminado: vacío. Se usa para leer. Debe estar en el formato Spark SQL.
- `QUERY`(String) - Predeterminado: vacío. Se usa para leer. Consulta SQL completa de Spark.
- `PARTITION_FIELD`(String): se usa para leer. Campo que se utilizará para particionar la consulta.
- `LOWER_BOUND`(Cadena): se usa para leer. Un valor de límite inferior inclusivo del campo de partición elegido.
- `UPPER_BOUND`(Cadena): se usa para leer. Un valor límite superior exclusivo del campo de partición elegido.

- `NUM_PARTITIONS`(Entero): predeterminado: 1. Se usa para leer. Número de particiones para leer.
- `IMPORT_DELETED_RECORDS`(Cadena) - Predeterminado: FALSO. Se usa para leer. Para obtener los registros eliminados durante la consulta.
- `WRITE_OPERATION`(String) - Predeterminado: INSERT. Se usa para escribir. El valor debe ser INSERT, UPDATE, UPSERT, DELETE.
- `ID_FIELD_NAMES`(String) - Predeterminado: nulo. Se usa solo para UPSERT.

Conexión Snowflake

Las siguientes propiedades se utilizan para configurar una conexión Snowflake que se utiliza en los trabajos de ETL. AWS Glue Al rastrear Snowflake, utilice una conexión JDBC.

URL de Snowflake

La URL del punto de conexión de Snowflake. Para obtener más información sobre las URL de los puntos de conexión de Snowflake, consulte Cómo conectarse a sus cuentas en la documentación de Snowflake.

AWS Secreto

El nombre secreto de un secreto en AWS Secrets Manager. AWS Glue se conectará a Snowflake usando las `sfPassword` claves `sfUser` y de tu secreto.

La función de Snowflake (opcional)

Se utilizará un rol AWS Glue de seguridad de Snowflake al conectarse.

Utilice las siguientes propiedades al configurar una conexión a un punto de conexión de Snowflake alojado en Amazon VPC mediante AWS PrivateLink.

VPC

Seleccione el nombre de la nube privada virtual (VPC) que contenga el almacén de datos. La consola de AWS Glue muestra todas las VPC para la región actual.

Subred

Seleccione la subred dentro de la VPC que contenga el almacén de datos. La consola de AWS Glue incluye todas las subredes para el almacén de datos en la VPC.

Grupos de seguridad

Elija los grupos de seguridad asociados a su almacén de datos. AWS Glue requiere uno o varios grupos de seguridad con una regla de origen de entrada que permite AWS Glue establezca la conexión. La consola de AWS Glue incluye todos los grupos de seguridad con acceso de entrada a su VPC. AWS Glue asocia estos grupos de seguridad con la interfaz de red elástica que se asocia a su subred de VPC.

Conexión vertical

Utilice las siguientes propiedades para configurar una conexión Vertica para AWS Glue los trabajos de ETL.

Host de Vertica

El nombre de host de su instalación de Vertica.

Puerto de Vertica

El puerto a través del cual está disponible su instalación de Vertica.

AWS Secreto

El nombre secreto de un secreto en AWS Secrets Manager. AWS Glue se conectará a Vertica usando las claves de tu secreto.

Utilice las siguientes propiedades al configurar una conexión a un punto de conexión de Vertica alojado en Amazon VPC.

VPC

Seleccione el nombre de la nube privada virtual (VPC) que contenga el almacén de datos. La consola de AWS Glue muestra todas las VPC para la región actual.

Subred

Seleccione la subred dentro de la VPC que contenga el almacén de datos. La consola de AWS Glue incluye todas las subredes para el almacén de datos en la VPC.

Grupos de seguridad

Elija los grupos de seguridad asociados a su almacén de datos. AWS Glue requiere uno o varios grupos de seguridad con una regla de origen de entrada que permite AWS Glue establezca la

conexión. La consola de AWS Glue incluye todos los grupos de seguridad con acceso de entrada a su VPC. AWS Glue asocia estos grupos de seguridad con la interfaz de red elástica que se asocia a su subred de VPC.

Conexión a SAP HANA

Utilice las siguientes propiedades para configurar una conexión SAP HANA para los trabajos de AWS Glue ETL.

URL DE SAP HANA

UNA URL DE SAP JDBC.

Las URL de JDBC de SAP HANA tienen el formato `jdbc:sap://saphanaHostname:saphanaPort/?databaseName=saphanaDBname,ParameterNa`

AWS Glue requiere los siguientes parámetros de URL de JDBC:

- `databaseName`: una base de datos predeterminada en SAP HANA a la que conectarse.

AWS Secreto

El nombre secreto de un secreto en AWS Secrets Manager. AWS Glue se conectará a SAP HANA mediante las claves de su secreto.

Utilice las siguientes propiedades al configurar una conexión a un punto de conexión de SAP HANA alojado en Amazon VPC:

VPC

Seleccione el nombre de la nube privada virtual (VPC) que contenga el almacén de datos. La consola de AWS Glue muestra todas las VPC para la región actual.

Subred

Seleccione la subred dentro de la VPC que contenga el almacén de datos. La consola de AWS Glue incluye todas las subredes para el almacén de datos en la VPC.

Grupos de seguridad

Elija los grupos de seguridad asociados a su almacén de datos. AWS Glue requiere uno o varios grupos de seguridad con una regla de origen de entrada que permite AWS Glue establezca la conexión. La consola de AWS Glue incluye todos los grupos de seguridad con acceso de entrada

a su VPC. AWS Glue asocia estos grupos de seguridad con la interfaz de red elástica que se asocia a su subred de VPC.

Conexión a Azure SQL

Utilice las siguientes propiedades para configurar una conexión SQL de Azure para los trabajos de AWS Glue ETL.

Dirección URL de Azure SQL

La dirección URL de JDBC de un punto de conexión de Azure SQL.

La lista de URL debe tener el siguiente formato: `jdbc:sqlserver://databaseServerName:databasePort;databaseName=azuresqlDBname;`.

AWS Glue requiere las siguientes propiedades de URL:

- `databaseName`: una base de datos predeterminada en Azure SQL a la que conectarse.

Para obtener más información sobre las direcciones URL de JDBC para instancias administradas de Azure SQL, consulte la documentación de Microsoft.

AWS Secreto

El nombre secreto de un secreto en AWS Secrets Manager. AWS Glue se conectará a Azure SQL mediante las claves de su secreto.

Conexión de Teradata Vantage

Utilice las siguientes propiedades para configurar una conexión Teradata Vantage para AWS Glue trabajos de ETL.

Dirección URL de Teradata

Para conectarse a una instancia de Teradata, especifique el nombre de host de la instancia de base de datos y los parámetros de Teradata pertinentes:

`jdbc:teradata://teradataHostname/ParameterName=ParameterValue,ParameterName=Pa`

AWS Glue admite los siguientes parámetros de URL de JDBC:

- `DATABASE_NAME`: una base de datos predeterminada de Teradata a la que conectarse.
- `DBS_PORT`: especifica el puerto de Teradata, si no es estándar.

AWS Secreto

El nombre secreto de un secreto en AWS Secrets Manager. AWS Glue se conectará a Teradata Vantage utilizando las claves de su secreto.

Utilice las siguientes propiedades al configurar una conexión a un punto de conexión de Teradata Vantage alojado en Amazon VPC:

VPC

Seleccione el nombre de la nube privada virtual (VPC) que contenga el almacén de datos. La consola de AWS Glue muestra todas las VPC para la región actual.

Subred

Seleccione la subred dentro de la VPC que contenga el almacén de datos. La consola de AWS Glue incluye todas las subredes para el almacén de datos en la VPC.

Grupos de seguridad

Elija los grupos de seguridad asociados a su almacén de datos. AWS Glue requiere uno o varios grupos de seguridad con una regla de origen de entrada que permite AWS Glue establezca la conexión. La consola de AWS Glue incluye todos los grupos de seguridad con acceso de entrada a su VPC. AWS Glue asocia estos grupos de seguridad con la interfaz de red elástica que se asocia a su subred de VPC.

OpenSearch Conexión de servicio

Utilice las siguientes propiedades para configurar una conexión de OpenSearch servicio para los trabajos de AWS Glue ETL.

Punto de conexión del dominio

Un punto final OpenSearch de dominio de Amazon Service tendrá el siguiente formulario predeterminado, https://search - *DomainName* -. *unstructuredIdContent región .es.amazonaws.com*. Para obtener más información sobre cómo identificar el punto de enlace de su dominio, consulte Creación y gestión de dominios de Amazon OpenSearch Service en la documentación OpenSearch de Amazon Service.

Puerto

El puerto abierto en el punto de conexión.

AWS Secreto

El nombre secreto de un secreto en AWS Secrets Manager. AWS Glue se conectará al OpenSearch Servicio mediante las claves de tu secreto.

Utilice las siguientes propiedades al configurar una conexión a un punto final OpenSearch de servicio alojado en Amazon VPC:

VPC

Seleccione el nombre de la nube privada virtual (VPC) que contenga el almacén de datos. La consola de AWS Glue muestra todas las VPC para la región actual.

Subred

Seleccione la subred dentro de la VPC que contenga el almacén de datos. La consola de AWS Glue incluye todas las subredes para el almacén de datos en la VPC.

Grupos de seguridad

Elija los grupos de seguridad asociados a su almacén de datos. AWS Glue requiere uno o varios grupos de seguridad con una regla de origen de entrada que permite AWS Glue establezca la conexión. La consola de AWS Glue incluye todos los grupos de seguridad con acceso de entrada a su VPC. AWS Glue asocia estos grupos de seguridad con la interfaz de red elástica que se asocia a su subred de VPC.

Conexión de Azure Cosmos

Utilice las siguientes propiedades para configurar una conexión de Azure Cosmos para los trabajos de AWS Glue ETL.

URI de punto de conexión de la cuenta de base de datos de Azure Cosmos DB

El punto de conexión usado para conectarse a Azure Cosmos. Para obtener más información, consulte la documentación de Azure.

AWS Secreto

El nombre secreto de un secreto en AWS Secrets Manager. AWS Glue se conectará a Azure Cosmos con las claves de su secreto.

Propiedades de las conexiones SSL de AWS Glue

A continuación, se muestran los detalles sobre la propiedad Require SSL connection (Solicitar conexión SSL).

Si no se solicita una conexión SSL, AWS Glue ignora los errores cuando utiliza SSL para cifrar una conexión a un almacén de datos. Para obtener instrucciones de configuración consulte la documentación del almacén de datos. Si se selecciona esta opción, se produce un error en la ejecución del trabajo, el rastreador o las instrucciones ETL de un punto de conexión de desarrollo cuando AWS Glue no puede conectarse.

> **Note**
>
> Snowflake admite una conexión SSL de forma predeterminada, por lo que esta propiedad no se aplica a Snowflake.

Esta opción se valida en el lado del cliente de AWS Glue. Para las conexiones JDBC, AWS Glue solo se conecta a través de SSL con validación de certificados y nombres de host. El soporte de conexión SSL está disponible para lo siguiente:

- Oracle Database
- Microsoft SQL Server
- PostgreSQL
- Amazon Redshift
- MySQL (solo instancias de Amazon RDS)
- Amazon Aurora MySQL (solo instancias de Amazon RDS)
- Amazon Aurora PostgreSQL (solo instancias de Amazon RDS)
- Kafka, que incluye Amazon Managed Streaming for Apache Kafka
- MongoDB

> **Note**
>
> Para habilitar un almacén de datos de Oracle de Amazon RDS para utilizar Solicitar conexión SSL, debe crear y asociar un grupo de opciones a la instancia de Oracle.

1. Inicie sesión en la consola de Amazon RDS AWS Management Console y ábrala en https://console.aws.amazon.com/rds/.
2. Agregue un Grupo de opciones a la instancia de Oracle de Amazon RDS. Para obtener más información sobre cómo agregar un grupo de opciones en la consola de Amazon RDS, consulte Creación de un grupo de opciones
3. Añada una Opción al grupo de opciones para SSL. El Puerto que especifique para SSL se usará posteriormente al crear una URL de conexión de JDBC de AWS Glue para la instancia de Oracle de Amazon RDS. Para obtener más información sobre cómo agregar una opción en la consola de Amazon RDS, consulte Agregar una opción a un grupo de opciones en la Guía del usuario de Amazon RDS. Para obtener más información acerca de las opciones de SSL de Oracle, consulte SSL de Oracle en la Guía del usuario de Amazon RDS.
4. En la consola de AWS Glue, cree una conexión a la instancia de Oracle de Amazon RDS. En la definición de conexión, seleccione Solicitar conexión SSL. Cuando se solicite, ingrese el Puerto que utilizó en la opción SSL de Oracle de Amazon RDS.

Las siguientes propiedades opcionales adicionales están disponibles cuando se selecciona Solicitar conexión SSL para una conexión:

Certificado JDBC personalizado en S3

Si tiene un certificado que utiliza actualmente para la comunicación SSL con sus bases de datos locales o en la nube, puede utilizar dicho certificado para las conexiones SSL a orígenes o destinos de datos de AWS Glue. Ingrese una ubicación de Amazon Simple Storage Service (Amazon S3) que contenga un certificado raíz personalizado. AWS Glue utiliza este certificado para establecer una conexión SSL a la base de datos. AWS Glue solo controla los certificados X.509. El certificado debe estar codificado en DER y suministrarse en formato PEM con codificación base64.

Si este campo se deja en blanco, se utiliza el certificado predeterminado.

Cadena de certificado JDBC personalizada

Ingrese la información del certificado específico de su base de datos JDBC. Esta cadena se utiliza para la coincidencia de dominios o la coincidencia de nombres distintivos (DN). Para Oracle Database, esta cadena se asigna al parámetro `SSL_SERVER_CERT_DN` de la sección de

seguridad del archivo `tnsnames.ora`. Para Microsoft SQL Server, esta cadena se utiliza como `hostNameInCertificate`.

A continuación se muestra un ejemplo del parámetro `SSL_SERVER_CERT_DN` de Oracle Database.

```
cn=sales,cn=OracleContext,dc=us,dc=example,dc=com
```

Ubicación del certificado de CA privada de Kafka

Si tiene un certificado que utiliza actualmente para la comunicación SSL con su almacén de datos de Kafka, puede utilizar dicho certificado con su conexión de AWS Glue. Esta opción es obligatoria para los almacenes de datos de Kafka y opcional para Amazon Managed Streaming for Apache Kafka los almacenes de datos. Ingrese una ubicación de Amazon Simple Storage Service (Amazon S3) que contenga un certificado raíz personalizado. AWS Glue utiliza este certificado para establecer una conexión SSL al almacén de datos de Kafka. AWS Glue solo controla los certificados X.509. El certificado debe estar codificado en DER y suministrarse en formato PEM con codificación base64.

Omitir la validación de certificados

Seleccione la casilla de verificación Omitir validación de certificado para omitir la validación del certificado personalizado por AWS Glue. Si decide validar, AWS Glue valida el algoritmo de firma y el algoritmo de clave pública de sujeto para el certificado. Si el certificado no supera la validación, se producirá un error en cualquier trabajo de ETL o rastreador que utilice la conexión.

Los únicos algoritmos de firma permitidos son SHA256withRSA, SHA384withRSA o SHA512withRSA. Para el algoritmo de clave pública del asunto, la longitud de la clave debe ser al menos 2048.

Ubicación del almacén de claves del cliente de Kafka

La ubicación de Amazon S3 del archivo de almacén de claves del cliente para la autenticación del lado del cliente Kafka. La ruta debe tener el formato s3://bucket/prefix/filename.jks. Debe terminar con el nombre de archivo y la extensión .jks.

Contraseña del almacén de claves del cliente de Kafka (opcional)

La contraseña para acceder al almacén de claves proporcionado.

Contraseña de la clave del cliente de Kafka (opcional)

Un almacén de claves puede consistir en varias claves, por lo que esta es la contraseña para acceder a la clave del cliente que se utilizará con la clave del lado del servidor Kafka.

Propiedades de las conexiones de Apache Kafka para la autenticación de clientes

AWS Glue admite el marco del nivel de seguridad y autenticación simples (SASL) para la autenticación cuando se crea una conexión de Apache Kafka. El esquema SASL es compatible con distintos mecanismos de autenticación; AWS Glue ofrece los protocolos SCRAM (nombre de usuario y contraseña), GSSAPI (protocolo Kerberos) y PLAIN.

Se utiliza AWS Glue Studio para configurar uno de los siguientes métodos de autenticación de clientes. Para obtener más información, consulte Creación de conexiones para conectores en la guía del AWS Glue Studio usuario.

- Ninguno: sin autenticación. Esto resulta útil si se crea una conexión con fines de prueba.
- SASL/SCRAM-SHA-512: la elección de este método de autenticación le permitirá especificar credenciales de autenticación. Existen dos opciones disponibles:
  - Usa AWS Secrets Manager (recomendado): si seleccionas esta opción, puedes almacenar tu nombre de usuario y contraseña en AWS Secrets Manager y permitir el AWS Glue acceso a ellos cuando lo necesites. Especificar el secreto donde están almacenadas las credenciales de autenticación SSL o SASL. Para obtener más información, consulte Almacenamiento de credenciales de conexión en AWS Secrets Manager.
  - Proporcione un nombre de usuario y una contraseña directamente.
- SASL/GSSAPI (Kerberos): si selecciona esta opción, puede seleccionar la ubicación del archivo keytab, el archivo krb5.conf e ingresar el nombre principal y el nombre del servicio de Kerberos. Las ubicaciones de los archivos keytab y krb5.conf deben estar en una ubicación de Simple Storage Service (Amazon S3). Dado que MSK aún no admite SASL/GSSAPI, esta opción solo está disponible para clústeres Apache Kafka administrados por el cliente. Para obtener más información, consulte MIT Kerberos Documentation: Keytab (Documentación de MIT Kerberos: Keytab).
- SASL/PLAIN: seleccione este método de autenticación para especificar las credenciales de autenticación. Existen dos opciones disponibles:

  - Usa AWS Secrets Manager (recomendado): si seleccionas esta opción, puedes almacenar tus credenciales en AWS Secrets Manager y permitir el AWS Glue acceso a la información cuando sea necesario. Especificar el secreto donde están almacenadas las credenciales de autenticación SSL o SASL.
  - Brinde de manera directa un usuario y una contraseña.
- Autenticación de cliente SSL: si selecciona esta opción, puede seleccionar la ubicación del almacén de claves del cliente Kafka navegando por Simple Storage Service (Amazon S3). Opcionalmente, puede ingresar la contraseña del almacén de claves del cliente Kafka y la contraseña de clave de cliente Kafka.

BigQuery Conexión a Google

Las siguientes propiedades se utilizan para configurar una BigQuery conexión de Google que se utiliza en los trabajos de AWS Glue ETL. Para obtener más información, consulte the section called “Conexiones de BigQuery”.

AWS Secreto

El nombre secreto de un secreto en AWS Secrets Manager. AWS Glue Los trabajos de ETL se conectarán a Google BigQuery mediante la `credentials` clave de tu secreto.

Conexión vertical

Las siguientes propiedades se utilizan para configurar una conexión Vertica utilizada en los trabajos de AWS Glue ETL. Para obtener más información, consulte the section called “Conexiones Vertica”.

Almacenamiento de credenciales de conexión en AWS Secrets Manager

Se recomienda utilizar AWS Secrets Manager para suministrar las credenciales de conexión del almacén de datos. Si se utiliza Secrets Manager de esta manera, se permite que AWS Glue acceda al secreto en el tiempo de ejecución para los trabajos de ETL y las ejecuciones de rastreadores, y ayuda a proteger las credenciales.

Requisitos previos

Para utilizar Secrets Manager con AWS Glue, se debe conceder al rol de IAM de AWS Glue permiso para recuperar valores de secretos. La política `AWSGlueServiceRole`, administrada por AWS, no incluye permisos de AWS Secrets Manager. Para ver ejemplos de políticas de IAM, consulte Ejemplo: Permiso para recuperar valores de secretos en la Guía del usuario de AWS Secrets Manager.

En función de la configuración de la red, es posible que también tenga que crear un punto de conexión de VPC para establecer una conexión privada entre la VPC y Secrets Manager. Para obtener más información, consulte Uso de un punto de conexión de VPC de AWS Secrets Manager.

Para crear un secreto para AWS Glue

1. Siga las instrucciones de Creación y administración de secretos de la Guía del usuario de AWS Secrets Manager. El siguiente ejemplo de JSON muestra cómo especificar las credenciales en la pestaña Plaintext (Texto sin formato) cuando se crea un secreto para AWS Glue.

   ```
   {
     "username": "EXAMPLE-USERNAME",
     "password": "EXAMPLE-PASSWORD"
   }
   ```

2. Asocie un secreto a una conexión mediante la interfaz de AWS Glue Studio. Para obtener instrucciones detalladas, consulte Creación de conexiones para conectores en la Guía del usuario de AWS Glue Studio.

Adición de una conexión de AWS Glue

Puede conectar a orígenes de datos en AWS Glue de Spark mediante programación. Para obtener más información, consulte Tipos de conexión y opciones para ETL en AWS Glue para Spark.

Puede usar la consola de AWS Glue para agregar, editar, eliminar y probar conexiones. Para obtener más información acerca de las conexiones a AWS Glue, consulte Conexión a datos.

Para agregar una conexión a AWS Glue

1. Inicie sesión en la AWS Glue consola AWS Management Console y ábrala en https://console.aws.amazon.com/glue/.
2. En el panel de navegación, en Data catalog (Catálogo de datos), elija Connections (Conexiones).

3. Elija Add connection (Agregar conexión) y, a continuación, complete el asistente, escribiendo las propiedades de conexión como se describe en the section called “Propiedades de las conexiones de AWS Glue”.

Conexión a Amazon Redshift en AWS Glue Studio

> **Note**
>
> Puedes usar Spark AWS Glue para leer y escribir en tablas de Amazon Redshift bases de datos externas a ellasAWS Glue Studio. Para configurar Amazon Redshift los AWS Glue trabajos mediante programación, consulteConexiones Redshift.

AWS Glueproporciona soporte integrado para. Amazon RedshiftAWS Glue Studioproporciona una interfaz visual a la que conectarse Amazon Redshift, crear trabajos de integración de datos y ejecutarlos en un entorno de ejecución de Spark AWS Glue Studio sin servidor.

Temas

- Crear una Amazon Redshift conexión
- Crear un nodo de origen de Amazon Redshift
- Crear un nodo Amazon Redshift de destino
- Opciones avanzadas

Crear una Amazon Redshift conexión

Permisos necesarios

Se necesitan permisos adicionales para usar Amazon Redshift clústeres y entornos Amazon Redshift sin servidor. Para obtener más información sobre cómo agregar permisos a los trabajos de ETL, consulte Revisar los permisos de IAM necesarios para los trabajos de ETL.

- corrimiento al rojo: DescribeClusters
- redshift, sin servidor: ListWorkgroups
- redshift, sin servidor: ListNamespaces

Información general

Al añadir una Amazon Redshift conexión, puede elegir una Amazon Redshift conexión existente o crear una nueva al añadir una fuente de datos (nodo Redshift) en. AWS Glue Studio

AWS Gluees compatible tanto con Amazon Redshift clústeres como con entornos Amazon Redshift sin servidor. Al crear una conexión, los entornos Amazon Redshift sin servidor muestran la etiqueta sin servidor junto a la opción de conexión.

Para obtener más información sobre cómo crear una Amazon Redshift conexión, consulte Mover datos de ida y vuelta. Amazon Redshift

Crear un nodo de origen de Amazon Redshift

Permisos necesarios

los trabajos de AWS Glue Studio que utilizan orígenes de datos de Amazon Redshift requieren permisos adicionales. Para obtener más información sobre cómo agregar permisos a los trabajos de ETL, consulte Revisar los permisos de IAM necesarios para los trabajos de ETL.

Se necesitan los siguientes permisos para utilizar una conexión de Amazon Redshift.

- redshift-data:ListSchemas
- redshift-data:ListTables
- redshift-data:DescribeTable
- redshift-data:ExecuteStatement
- redshift-data:DescribeStatement
- redshift-data:GetStatementResult

Agregar un origen de datos de Amazon Redshift

Para agregar un nodo de Origen de datos: Amazon Redshift:

1. Elija el tipo de acceso a Amazon Redshift:
    - Conexión de datos directa (recomendada): elija esta opción si desea acceder a sus datos de Amazon Redshift directamente. Esta es la opción recomendada y también la predeterminada.
    - Data Catalog tables— Elija esta opción si hay tablas del catálogo de datos que desee utilizar.

2. Si elige Conexión de datos directa, elija la conexión para el origen de datos de Amazon Redshift. Esto supone que la conexión ya existe y que puede seleccionar entre las conexiones existentes. Si necesita crear una conexión, elija Crear conexión de Redshift. Para más información, consulte Información general sobre el uso de conectores y conexiones.

   Una vez que haya elegido una conexión, puede ver las propiedades de la conexión mediante un clic en Ver propiedades. La información sobre la conexión está visible, como la URL, los grupos de seguridad, la subred, la zona de disponibilidad, la descripción y las marcas horarias creadas (UTC) y actualizadas por última vez (UTC).

3. Elija una opción de origen de Amazon Redshift:

   - Elija una sola tabla: esta es la tabla que contiene los datos a los que desea acceder desde una sola tabla de Amazon Redshift.
   - Ingresar una consulta personalizada: permite acceder a un conjunto de datos de varias tablas de Amazon Redshift en función del consulta personalizada.

4. Si eligió una sola tabla, elija el esquema de Amazon Redshift. La lista de esquemas disponibles para elegir se determina por la tabla seleccionada.

   O bien, elija Ingresar consulta personalizada. Elija esta opción para acceder a un conjunto de datos personalizado desde varias tablas de Amazon Redshift. Al elegir esta opción, ingrese la consulta de Amazon Redshift.

   Al conectarse a un entorno de Amazon Redshift sin servidor, agregue el siguiente permiso a la consulta personalizada:

   ```
   GRANT SELECT ON ALL TABLES IN <schema> TO PUBLIC
   ```

   Puede elegir Deducir el esquema para leer el esquema en función de la consulta que haya introducido. También puede elegir el editor de consultas Open Redshift para ingresar una consulta de Amazon Redshift. Para obtener más información, vea Consulta de una base de datos mediante el editor de consultas.

5. En Rendimiento y seguridad, elija el directorio provisional de Amazon S3 y el rol de IAM.

   - Directorio de almacenamiento provisional de Amazon S3: elija la ubicación de Amazon S3 para almacenar los datos temporalmente.

- Rol de IAM: elija la función de IAM que pueda escribir en la ubicación de Amazon S3 que haya seleccionado.

6. En Parámetros de Redshift personalizados (opcional), ingrese el parámetro y el valor.

Crear un nodo Amazon Redshift de destino

Permisos necesarios

AWS Glue Studio los trabajos que utilizan un destino de Amazon Redshift datos requieren permisos adicionales. Para obtener más información sobre cómo agregar permisos a los trabajos de ETL, consulte Revisar los permisos de IAM necesarios para los trabajos de ETL.

Se necesitan los siguientes permisos para utilizar una Amazon Redshift conexión.

- datos de redshift: ListSchemas
- datos de desplazamiento al rojo: ListTables

Añadir un nodo de destino Amazon Redshift

Para crear un nodo Amazon Redshift de destino:

1. Elija una Amazon Redshift tabla existente como destino o introduzca un nombre de tabla nuevo.
2. Al utilizar el nodo destino de datos: Redshift, puede elegir entre las siguientes opciones:

   - APPEND: si ya existe una tabla, coloque todos los datos nuevos en la tabla como un inserto. Si la tabla no existe, créela y, a continuación, inserte todos los datos nuevos.

     Además, seleccione la casilla si desea actualizar (UPSERT) los registros existentes en la tabla de destino. La tabla debe existir primero; de lo contrario, la operación fallará.
   - MERGE: AWS Glue actualizará o anexará datos a la tabla de destino en función de las condiciones que especifique.

     > **Note**
     >
     > Para utilizar la acción de combinaciónAWS Glue, debe activar la función de Amazon Redshift combinación. Para obtener instrucciones sobre cómo habilitar la fusión en tu Amazon Redshift instancia, consulta MERGE (vista previa).

Elija opciones:

- Elegir claves y acciones sencillas: elija las columnas que se usarán como claves de coincidencia entre los orígenes de datos y el conjunto de destinos de datos.

  Especifique las siguientes opciones cuando coincidan:

  - Actualice el registro del conjunto de datos de destino con los datos de origen.
  - Elimine el registro del conjunto de datos de destino.

  Especifique las siguientes opciones cuando no coincidan:

  - Inserte los datos de origen como una nueva fila en el conjunto de datos de destino.
  - No hacer nada.
- Ingrese una instrucción MERGE personalizada: a continuación, puede elegir Validar la instrucción MERGE para comprobar si la instrucción es válida o no.

- TRUNCATE: si ya existe una tabla, trunque los datos de la tabla al borrar primero el contenido de la tabla de destino. Si el truncado se realiza correctamente, inserte todos los datos. Si la tabla no existe, créela y, a continuación, inserte todos los datos. Si el truncado no es exitoso, la operación producirá un error.
- DROP: si una tabla ya existe, elimine los metadatos y los datos de la tabla. Si el borrado se realiza correctamente, inserte todos los datos. Si la tabla no existe, créela y, a continuación, inserte todos los datos. Si el descarte no es exitoso, la operación producirá un error.
- CREATE: cree una tabla nueva con el nombre predeterminado. Si el nombre de la tabla ya existe, cree una nueva tabla con un sufijo de nombre `job_datetime` en el nombre para que sea único. Esto insertará todos los datos en la nueva tabla. Si la tabla existe, el nombre final de la tabla tendrá el sufijo adjunto. Si la tabla no existe, se creará una tabla. En cualquier caso, se creará una tabla nueva.

Opciones avanzadas

Consulte Uso del conector Spark de Amazon Redshift en AWS Glue.

Conexión a Azure Cosmos DB en AWS Glue Studio

AWS Glue proporciona soporte integrado para Azure Cosmos DB. AWS Glue Studio proporciona una interfaz visual para conectarse a Azure Cosmos DB for NoSQL, crear trabajos de integración de datos y ejecutarlos en el tiempo de ejecución de Spark sin servidor de AWS Glue Studio.

Temas

- Creación de una conexión a Azure Cosmos DB
- Creación de un nodo de origen de Azure Cosmos DB
- Creación de un nodo de destino de Azure Cosmos DB
- Opciones avanzadas

Creación de una conexión a Azure Cosmos DB

Requisitos previos:

- En Azure, tendrá que identificar o generar una clave de Azure Cosmos DB para que la utilice AWS Glue, `cosmosKey`. Para obtener más información, consulte Acceso seguro a los datos en Azure Cosmos DB en la documentación de Azure.

Para configurar una conexión a Azure Cosmos DB:

1. En AWS Secrets Manager, cree un secreto con su clave de datos de Azure Cosmos DB. Para crear un secreto en Secrets Manager, siga el tutorial disponible en Crear un secreto AWS Secrets Manager en la documentación AWS Secrets Manager. Después de crear el secreto, conserve el nombre secreto, *`secretName`*, para el siguiente paso.
   - Al seleccionar pares clave/valor, genere un par para la clave `spark.cosmos.accountKey` con el valor *`cosmosKey`*.
2. En la consola de AWS Glue, genere una conexión mediante los pasos que se indican en the section called “Adición de una conexión de AWS Glue”. Tras crear la conexión, conserve el nombre de la conexión, *`connectionName`*, para el uso futuro en AWS Glue.
   - Al seleccionar un tipo de conexión, seleccione Azure Cosmos DB.
   - Al seleccionar un secreto AWS, proporcione un *`secretName`*.

Creación de un nodo de origen de Azure Cosmos DB

Requisitos previos necesarios

- Una conexión a AWS Glue Azure Cosmos DB, configurada con un AWS Secrets Manager secreto, tal y como se describe en la sección anterior, the section called “Creación de una conexión a Azure Cosmos DB”.
- Permisos adecuados en el trabajo para leer el secreto utilizado por la conexión.
- Un contenedor de Azure Cosmos DB para NoSQL del que desearía leer. Necesitará la información de identificación del contenedor.

  Un contenedor de Azure Cosmos para NoSQL se identifica por su base de datos y su contenedor. Debe proporcionar los nombres de la base de datos, `cosmosDBName`, y del contenedor, `cosmosContainerName`, al conectarse a la API de Azure Cosmos para NoSQL.

Agregar un origen de datos de Azure Cosmos DB

Para agregar un nodo de Origen de datos - Azure Cosmos DB:

1. Elija la conexión para el origen de datos de Azure Cosmos DB. Dado que lo ha creado, debería estar disponible en el menú desplegable. Si es necesario crear una conexión, elija Crear conexión de Azure Cosmos DB. Para obtener más información, consulte la sección the section called “Creación de una conexión a Azure Cosmos DB” anterior.

   Una vez que haya elegido una conexión, puede ver las propiedades de la conexión mediante un clic en Ver propiedades.
2. Elija el Nombre de la base de datos de Cosmos DB: proporcione el nombre de la base de datos desde la que desea leer, `CosmosDBName`.
3. Elija Azure Cosmos DB Container: proporcione el nombre del contenedor desde el que desea leer, `CosmosContainerName`.
4. Si lo desea, elija Azure Cosmos DB Custom Query: proporcione una consulta SQL SELECT para recuperar información específica de Azure Cosmos DB.
5. En Propiedades personalizadas de Azure Cosmos, ingrese los parámetros y valores según sea necesario.

Creación de un nodo de destino de Azure Cosmos DB

Requisitos previos necesarios

- Una conexión a AWS Glue Azure Cosmos DB, configurada con un AWS Secrets Manager secreto, tal y como se describe en la sección anterior, the section called “Creación de una conexión a Azure Cosmos DB”.
- Permisos adecuados en el trabajo para leer el secreto utilizado por la conexión.
- Una tabla de Azure Cosmos DB a la que desearía escribir. Necesitará la información de identificación del contenedor. Debe crear el contenedor antes de llamar al método de conexión.

  Un contenedor de Azure Cosmos para NoSQL se identifica por su base de datos y su contenedor. Debe proporcionar los nombres de la base de datos, *cosmosDBName*, y del contenedor, *cosmosContainerName*, al conectarse a la API de Azure Cosmos para NoSQL.

Agregar un destino de datos de Azure Cosmos DB

Para agregar un nodo de Destino de datos - Azure Cosmos DB:

1. Elija la conexión para el origen de datos de Azure Cosmos DB. Dado que lo ha creado, debería estar disponible en el menú desplegable. Si es necesario crear una conexión, elija Crear conexión de Azure Cosmos DB. Para obtener más información, consulte la sección the section called “Creación de una conexión a Azure Cosmos DB” anterior.

   Una vez que haya elegido una conexión, puede ver las propiedades de la conexión mediante un clic en Ver propiedades.
2. Elija el Nombre de la base de datos de Cosmos DB: proporcione el nombre de la base de datos desde la que desea leer, *CosmosDBName*.
3. Elija Azure Cosmos DB Container: proporcione el nombre del contenedor desde el que desea leer, *CosmosContainerName*.
4. En Propiedades personalizadas de Azure Cosmos, ingrese los parámetros y valores según sea necesario.

Opciones avanzadas

Puede brindar opciones avanzadas al crear un nodo de Azure Cosmos DB. Estas opciones son las mismas que las disponibles cuando se programa AWS Glue para scripts de Spark.

Consulte the section called “Conexiones de Azure Cosmos DB”.

Conexión a Azure SQL en AWS Glue Studio

AWS Glue proporciona soporte integrado para Azure SQL. AWS Glue Studio proporciona una interfaz visual para conectarse a Azure SQL, crear trabajos de integración de datos y ejecutarlos en el tiempo de ejecución de Spark sin servidor de AWS Glue Studio.

Temas

- Creación de una conexión de Azure SQL
- Creación de un nodo de origen de Azure SQL
- Creación de un nodo de destino de Azure SQL
- Opciones avanzadas

Creación de una conexión de Azure SQL

Para conectarse a Azure SQL desde AWS Glue, tendrá que crear y almacenar sus credenciales de Azure SQL en un AWS Secrets Manager secreto y, a continuación, asociar ese secreto a una conexión AWS Glue de Azure SQL.

Para configurar una conexión a Azure SQL:

1. En AWS Secrets Manager, cree un secreto con sus credenciales de Azure SQL. Para crear un secreto en Secrets Manager, siga el tutorial disponible en Crear un secreto AWS Secrets Manager en la documentación AWS Secrets Manager. Después de crear el secreto, conserve el nombre secreto, *`secretName`*, para el siguiente paso.
   - Al seleccionar pares clave/valor, genere un par para la clave `user` con el valor *`azuresqlUsername`*.
   - Al seleccionar pares clave/valor, genere un par para la clave `password` con el valor *`azuresqlPassword`*.
2. En la consola de AWS Glue, genere una conexión mediante los pasos que se indican en the section called “Adición de una conexión de AWS Glue”. Tras crear la conexión, conserve el nombre de la conexión, *`connectionName`*, para el uso futuro en AWS Glue.
   - Al seleccionar un tipo de conexión, seleccione Azure SQL.
   - Al proporcionar una URL de Azure SQL, proporcione una URL de punto de conexión de JDBC.

La lista de URL debe tener el siguiente formato: `jdbc:sqlserver://databaseServerName:databasePort;databaseName=azuresqlDBnam`

AWS Glue requiere las siguientes propiedades de URL:

- `databaseName`: una base de datos predeterminada en Azure SQL a la que conectarse.

Para obtener más información sobre las direcciones URL de JDBC para instancias administradas de Azure SQL, consulte la documentación de Microsoft.

- Al seleccionar un secreto AWS, proporcione un *secretName*.

Creación de un nodo de origen de Azure SQL

Requisitos previos necesarios

- Una conexión de AWS Glue Azure SQL, configurada con un AWS Secrets Manager secreto, como se describe en la sección anterior, the section called “Creación de una conexión de Azure SQL”.
- Permisos adecuados en el trabajo para leer el secreto utilizado por la conexión.
- Una tabla de Azure SQL de la que quiera leer, *tableName*.

Una tabla de Azure SQL se identifica por su base de datos, esquema y nombre de tabla. Debe proporcionar el nombre de la base de datos y el nombre de la tabla al conectarse a Azure SQL. También debe proporcionar el esquema si no es el predeterminado, "público". La base de datos se proporciona a través de una propiedad URL en *connectionName*, y el nombre de la tabla y el esquema a través de `dbtable`.

Agregar un origen de datos de Azure SQL

Para agregar un nodo de Origen de datos - Azure SQL:

1. Elija la conexión para el origen de datos de Azure SQL. Dado que lo ha creado, debería estar disponible en el menú desplegable. Si es necesario crear una conexión, elija Crear conexión de Azure SQL. Para obtener más información, consulte la sección the section called “Creación de una conexión de Azure SQL” anterior.

   Una vez que haya elegido una conexión, puede ver las propiedades de la conexión mediante un clic en Ver propiedades.
2. Elija una opción de Origen de Azure SQL:

- Elija una sola tabla: acceda a todos los datos de una sola tabla.
- Ingresar una consulta personalizada: permite acceder a un conjunto de datos de varias tablas en función del consulta personalizada.

3. Si eligió una sola tabla, ingrese *tableName*.

   Si eligió Introducir una consulta personalizada, introduzca una consulta SELECT de TransactSQL.

4. En Propiedades personalizadas de Azure SQL, ingrese los parámetros y valores según sea necesario.

Creación de un nodo de destino de Azure SQL

Requisitos previos necesarios

- Una conexión de AWS Glue Azure SQL, configurada con un AWS Secrets Manager secreto, como se describe en la sección anterior, the section called “Creación de una conexión de Azure SQL”.
- Permisos adecuados en el trabajo para leer el secreto utilizado por la conexión.
- Una tabla de Azure SQL a la que desearía escribir, *tableName*.

  Una tabla de Azure SQL se identifica por su base de datos, esquema y nombre de tabla. Debe proporcionar el nombre de la base de datos y el nombre de la tabla al conectarse a Azure SQL. También debe proporcionar el esquema si no es el predeterminado, "público". La base de datos se proporciona a través de una propiedad URL en *connectionName*, y el nombre de la tabla y el esquema a través de `dbtable`.

Agregar un destino de datos de Azure SQL

Para agregar un nodo de Destino de datos - Azure SQL:

1. Elija la conexión para el origen de datos de Azure SQL. Dado que lo ha creado, debería estar disponible en el menú desplegable. Si es necesario crear una conexión, elija Crear conexión de Azure SQL. Para obtener más información, consulte la sección the section called “Creación de una conexión de Azure SQL” anterior.

   Una vez que haya elegido una conexión, puede ver las propiedades de la conexión mediante un clic en Ver propiedades.

2. Configure el nombre de la tabla proporcionando *`tableName`*.
3. En Propiedades personalizadas de Azure SQL, ingrese los parámetros y valores según sea necesario.

Opciones avanzadas

Puede brindar opciones avanzadas al crear un nodo de Azure SQL. Estas opciones son las mismas que las disponibles cuando se programa AWS Glue para scripts de Spark.

Consulte the section called “Conexiones SQL Azure”.

Conectarse a Google BigQuery en AWS Glue Studio

> **Note**
>
> Puedes usar Spark AWS Glue para leer y escribir en tablas de Google BigQuery en la versión AWS Glue 4.0 y versiones posteriores. Para configurar Google BigQuery con AWS Glue tareas mediante programación, consulta Conexiones de BigQuery.

AWS Glue Studioproporciona una interfaz visual a la que conectarse BigQuery, crear trabajos de integración de datos y ejecutarlos en el entorno de ejecución AWS Glue Studio sin servidor de Spark.

Temas

- Crear una conexión con BigQuery
- Crear un nodo de origen de BigQuery
- Creación de un nodo de destino de BigQuery
- Opciones avanzadas

Crear una conexión con BigQuery

Para conectarse a Google BigQuery desde AWS Glue, tendrá que generar y almacenar las credenciales de Google Cloud Platform en un secreto de AWS Secrets Manager y, a continuación, asociar ese secreto a una conexión de Google BigQuery AWS Glue.

Para configurar una conexión a BigQuery, siga estos pasos:

1. En Google Cloud Platform, genere e identifique los recursos pertinentes:

   - Genere e identifique un proyecto de GCP que contenga tablas de BigQuery a las que desea conectarse.
   - Habilite la API de BigQuery. Para obtener más información, consulte Cómo usar la API de lectura de almacenamiento de BigQuery para leer datos de tablas.

2. En Google Cloud Platform, genere y exporte las credenciales de la cuenta de servicio:

   Puede usar el asistente de credenciales de BigQuery para simplificar este paso: Crear credenciales.

   Para crear una cuenta de servicio en GCP, siga el tutorial disponible en Crear cuentas de servicio.

   - Al elegir un proyecto, seleccione el proyecto que contiene su tabla de BigQuery.
   - Cuando elija los roles de IAM de GCP para la cuenta de servicio, agregue o genere una función que conceda los permisos apropiados para ejecutar los trabajos en BigQuery, como la lectura, escritura o creación de tablas de BigQuery.

   Para crear credenciales para la cuenta de servicio, siga el tutorial disponible en Crear una clave de cuenta de servicio.

   - Al seleccionar el tipo de clave, seleccione JSON.

   Ahora debería haber descargado un archivo JSON que contiene las credenciales de la cuenta de servicio. Debería parecerse a lo que sigue:

```
{
  "type": "service_account",
  "project_id": "*****",
  "private_key_id": "*****",
  "private_key": "*****",
  "client_email": "*****",
  "client_id": "*****",
  "auth_uri": "https://accounts.google.com/o/oauth2/auth",
  "token_uri": "https://oauth2.googleapis.com/token",
  "auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs",
```

```
  "client_x509_cert_url": "*****",
  "universe_domain": "googleapis.com"
}
```

3. Codifique en base64 el archivo de credenciales descargado. En una sesión AWS CloudShell o similar, puede hacerlo desde la línea de comandos al ejecutar `cat` *`credentialsFile.json`* `| base64 -w 0`. Conserve el resultado de este comando, *`credentialString`*.
4. En AWS Secrets Manager, genere un secreto mediante las credenciales de la plataforma de Google Cloud. Para crear un secreto en Secrets Manager, siga el tutorial disponible en Crear un secreto AWS Secrets Manager en la documentación AWS Secrets Manager. Después de crear el secreto, conserve el nombre secreto, *`secretName`*, para el siguiente paso.
   - Al seleccionar pares clave/valor, genere un par para la clave `credentials` con el valor *`credentialString`*.
5. En el Catálogo de datos de AWS Glue, genere una conexión mediante los pasos que se indican en https://docs.aws.amazon.com/glue/latest/dg/console-connections.html. Tras crear la conexión, conserve el nombre de la conexión, *`connectionName`*, para el siguiente paso.
   - Cuando elija un Tipo de conexión, seleccione Google BigQuery.
   - Al seleccionar un secreto AWS, proporcione un *`secretName`*.
6. Otorga permiso al rol de IAM asociado al trabajo de AWS Glue para leer el *`secretName`*.
7. En la configuración del trabajo de AWS Glue, proporcione un *`connectionName`* como una Conexión de red adicional.

Crear un nodo de origen de BigQuery

Requisitos previos necesarios

- Una conexión del Catálogo de datos de AWS Glue tipo BigQuery
- Un secreto de AWS Secrets Manager de las credenciales de Google BigQuery, que utiliza la conexión.
- Permisos adecuados en el trabajo para leer el secreto utilizado por la conexión.
- El nombre y el conjunto de datos de la tabla y el proyecto de Google Cloud correspondiente que quiere leer.

Agregar un origen de datos de BigQuery

Para agregar un nodo de Origen de datos: BigQuery:

1. Elija la conexión para el origen de datos de BigQuery. Dado que lo ha creado, debería estar disponible en el menú desplegable. Si es necesario crear una conexión, elija Crear conexión de BigQuery. Para más información, consulte Información general sobre el uso de conectores y conexiones.

   Una vez que haya elegido una conexión, puede ver las propiedades de la conexión mediante un clic en Ver propiedades.
2. Identifique qué datos de BigQuery desea leer y, a continuación, elija una opción de origen de BigQuery

   - Elegir una sola tabla: permite extraer todos los datos de una tabla.
   - Introducir una consulta personalizada: permite personalizar los datos que se van a recuperar mediante una consulta.
3. Describa los datos que desea leer

   (Obligatorio) defina el Proyecto principal como el proyecto que contiene la tabla o un proyecto principal de facturación, si corresponde.

   Si ha elegido una sola tabla, establezca Tabla con el nombre de una tabla de Google BigQuery en el siguiente formato: `[dataset].[table]`

   Si ha elegido una consulta, envíela a Consulta. En su consulta, consulte las tablas con su nombre de tabla completo, en el formato: `[project].[dataset].[tableName]`.
4. Proporcionar las propiedades de BigQuery

   Si ha elegido una sola tabla, no será necesario proporcionar propiedades adicionales.

   Si eligió una consulta, debe proporcionar las siguientes Propiedades personalizadas de Google BigQuery:

   - Establezca `viewsEnabled` en true.
   - Establezca `materializationDataset` en un conjunto de datos. La entidad principal de GCP autenticada por las credenciales proporcionadas a través de la conexión de AWS Glue debe poder crear tablas en este conjunto de datos.

Creación de un nodo de destino de BigQuery

Requisitos previos necesarios

- Una conexión del Catálogo de datos de AWS Glue tipo BigQuery
- Un secreto de AWS Secrets Manager de las credenciales de Google BigQuery, que utiliza la conexión.
- Permisos adecuados en el trabajo para leer el secreto utilizado por la conexión.
- El nombre y el conjunto de datos de la tabla y el proyecto de Google Cloud correspondiente al que desea escribir.

Agregar un destino de datos de BigQuery

Para agregar un nodo de Destino de datos: BigQuery, siga estos pasos:

1. Elija la conexión para el destino de datos de BigQuery. Dado que lo ha creado, debería estar disponible en el menú desplegable. Si es necesario crear una conexión, elija Crear conexión de BigQuery. Para más información, consulte Información general sobre el uso de conectores y conexiones.

   Una vez que haya elegido una conexión, puede ver las propiedades de la conexión mediante un clic en Ver propiedades.
2. Identifique en qué tabla de BigQuery desea escribir y, a continuación, elija un Método de escritura.

   - Directo: escribe en BigQuery directamente mediante la API de escritura de BigQuery Storage.
   - Indirecto: escribe en Google Cloud Storage y, luego, copia en BigQuery.

   Si quiere escribir de forma indirecta, proporcione una ubicación de GCS de destino con un bucket de GCS temporal. Deberá proporcionar una configuración adicional en la conexión de AWS Glue. Para obtener más información, consulte Uso de escritura indirecta con Google BigQuery.
3. Describa los datos que desea leer

   (Obligatorio) defina el Proyecto principal como el proyecto que contiene la tabla o un proyecto principal de facturación, si corresponde.

Si ha elegido una sola tabla, establezca Tabla con el nombre de una tabla de Google BigQuery en el siguiente formato: `[dataset].[table]`

Opciones avanzadas

Puedes proporcionar opciones avanzadas al crear un BigQuery nodo. Estas opciones son las mismas que las disponibles al programar AWS Glue para scripts de Spark.

Consulta BigQuery la referencia sobre las opciones de conexión en la guía para AWS Glue desarrolladores.

Conexión a MongoDB en AWS Glue Studio

AWS Glue proporciona soporte integrado para MongoDB. AWS Glue Studio proporciona una interfaz visual para conectarse a MongoDB, crear trabajos de integración de datos y ejecutarlos en el tiempo de ejecución de Spark sin servidor de AWS Glue Studio.

Temas

- Creación de una conexión de MongoDB
- Creación de un nodo de origen de MongoDB
- Creación de un nodo de destino de MongoDB
- Opciones avanzadas

Creación de una conexión de MongoDB

Requisitos previos:

- Si su instancia de MongoDB está en una Amazon VPC, configure Amazon VPC para permitir que su trabajo de AWS Glue se comunique con la instancia de MongoDB sin que el tráfico atraviese la Internet pública.

  En Amazon VPC, identifique o cree una VPC, una subred y un grupo de seguridad que AWS Glue utilizará al ejecutar el trabajo. Además, debe asegurarse de que Amazon VPC esté configurada para permitir el tráfico de red entre su instancia de MongoDB y esta ubicación. Según el diseño de la red, esto puede requerir cambios en las reglas de los grupos de seguridad, las ACL de red, las puertas de enlace de NAT y las conexiones entre pares.

Para configurar una conexión a MongoDB:

1. Si lo desea, en AWS Secrets Manager, cree un secreto con sus credenciales de MongoDB. Para crear un secreto en Secrets Manager, siga el tutorial disponible en Crear un secreto AWS Secrets Manager en la documentación AWS Secrets Manager. Después de crear el secreto, conserve el nombre secreto, *secretName*, para el siguiente paso.

   - Al seleccionar pares clave/valor, genere un par para la clave `username` con el valor *mongodbUser*.

     Al seleccionar pares clave/valor, genere un par para la clave `password` con el valor *mongodbPass*.

2. En la consola de AWS Glue, genere una conexión mediante los pasos que se indican en the section called “Adición de una conexión de AWS Glue”. Tras crear la conexión, conserve el nombre de la conexión, *connectionName*, para el uso futuro en AWS Glue.

   - Al seleccionar un tipo de conexión, seleccione MongoDB o MongoDB Atlas.
   - Al seleccionar la URL de MongoDB o la URL de MongoDB Atlas, proporciona el nombre de host de la instancia de MongoDB.

     Se proporciona una URL de MongoDB en este formato `mongodb://`*mongoHost*`:`*mongoPort*`/`*mongoDBname*.

     Se proporciona una URL de MongoDB Atlas en este formato `mongodb+srv://`*mongoHost*`:`*mongoPort*`/`*mongoDBname*.

     Proporcionar la base de datos predeterminada para la conexión, *mongoDBname* es opcional.
   - Si eligió crear un secreto de Secrets Manager, elija el tipo de credencial AWS Secrets Manager.

     Luego, en AWS Secret, ingrese *secretName*.
   - Si decide proporcionar el nombre de usuario y la contraseña, proporcione *mongoDBuser* y *mongodbPass*.

3. En las siguientes situaciones, es posible que necesite una configuración adicional:

   - Para instancias de MongoDB alojadas AWS en una VPC de Amazon

- Deberá proporcionar la información de conexión de Amazon VPC a la conexión AWS Glue que define sus credenciales de seguridad de MongoDB. Al crear o actualizar la conexión, configure los VPC, Subred y los grupos de seguridad en Opciones de red.

Tras crear una conexión con AWS Glue MongoDB, deberá realizar los siguientes pasos antes de ejecutar su trabajo de AWS Glue:

- Cuando trabaje con trabajos AWS Glue en el editor visual, debe proporcionar la información de conexión de Amazon VPC para que su trabajo se conecte a MongoDB. Identifique una ubicación adecuada en Amazon VPC y envíela a su conexión de AWS Glue MongoDB.
- Si opta por crear un secreto de Secrets Manager, conceda permiso al rol de IAM asociado a su trabajo de AWS Glue para leer *`secretName`*.

Creación de un nodo de origen de MongoDB

Requisitos previos necesarios

- Una conexión AWS Glue MongoDB, como se describe en la sección anterior,. the section called “Creación de una conexión de MongoDB”
- Si eligió crear un secreto de Secrets Manager, asigne los permisos necesarios para leer el secreto utilizado por la conexión.
- Una colección de MongoDB de la que quiera leer. Necesitará información de identificación para la colección.

  Una colección MongoDB se identifica mediante un nombre de base de datos y un nombre de colección, *`mongodbName`*, *`mongodbCollection`*.

Agregar un origen de datos de MongoDB

Para agregar un nodo de Origen de datos - MongoDB:

1. Elija la conexión para el origen de datos de MongoDB. Dado que lo ha creado, debería estar disponible en el menú desplegable. Si es necesario crear una conexión, elija Crear conexión de MongoDB. Para obtener más información, consulte la sección the section called “Creación de una conexión de MongoDB” anterior.

Una vez que haya elegido una conexión, puede ver las propiedades de la conexión mediante un clic en Ver propiedades.

2. Elija una Base de datos. Introduzca *mongodbName.*
3. Elija una Colección. Ingrese a *mongodbCollection.*
4. Elija el Particionador, el tamaño de la partición (MB) y la Clave de partición. Para obtener más información sobre los parámetros de particiones, consulte the section called “"connectionType": "mongodb" como origen”.
5. En Propiedades personalizadas de MongoDB, ingrese los parámetros y valores según sea necesario.

Creación de un nodo de destino de MongoDB

Requisitos previos necesarios

- Una conexión AWS Glue MongoDB, configurada con un AWS Secrets Manager secreto, como se describe en la sección anterior, the section called “Creación de una conexión de MongoDB”.
- Permisos adecuados en el trabajo para leer el secreto utilizado por la conexión.
- Una tabla de MongoDB a la que desearía escribir, *tableName*.

Agregar un destino de datos de MongoDB

Para añadir un nodo de Destino de datos - MongoDB:

1. Elija la conexión para el origen de datos de MongoDB. Dado que lo ha creado, debería estar disponible en el menú desplegable. Si es necesario crear una conexión, elija Crear conexión de MongoDB. Para obtener más información, consulte la sección the section called “Creación de una conexión de MongoDB” anterior.

   Una vez que haya elegido una conexión, puede ver las propiedades de la conexión mediante un clic en Ver propiedades.
2. Elija una Base de datos. Introduzca *mongodbName.*
3. Elija una Colección. Ingrese a *mongodbCollection.*
4. Elija el Particionador, el tamaño de la partición (MB) y la Clave de partición. Para obtener más información sobre los parámetros de particiones, consulte the section called “"connectionType": "mongodb" como origen”.

5. Seleccione Reintentar escrituras si lo desea.
6. En Propiedades personalizadas de MongoDB, ingrese los parámetros y valores según sea necesario.

Opciones avanzadas

Puede brindar opciones avanzadas al crear un nodo de MongoDB. Estas opciones son las mismas que las disponibles cuando se programa AWS Glue para scripts de Spark.

Consulte the section called “Conexión a MongoDB”.

Conexión al servicio OpenSearch en AWS Glue Studio

AWS Glue proporciona soporte integrado para Amazon OpenSearch Service. AWS Glue Studio proporciona una interfaz visual para conectarse a Amazon OpenSearch Service, crear trabajos de integración de datos y ejecutarlos en el tiempo de ejecución de Spark sin servidor de AWS Glue Studio. Esta característica no es compatible con OpenSearch Service sin servidor.

Temas

- Crear una conexión con OpenSearch Service
- Crear un nodo fuente de OpenSearch Service
- Crear un nodo de destino de OpenSearch Service
- Opciones avanzadas

Crear una conexión con OpenSearch Service

Requisitos previos:

- Identifique el punto de conexión del dominio, el *AOSEndpoint* y el puerto, *AOSport* desde el que desea leer o cree el recurso siguiendo las instrucciones de la documentación de Amazon OpenSearch Service. Para obtener más información sobre la creación de un dominio, consulte Crear y administrar dominios de Amazon OpenSearch Service en la documentación de Amazon OpenSearch Service.

  Un punto de conexión de dominio de Amazon OpenSearch Service tendrá el siguiente formulario predeterminado: https://search-*domainName*-*unstructuredIdContent*.*region*.es.amazonaws.com. Para obtener más

información sobre cómo identificar su punto de conexión de dominio, consulte Crear y administrar dominios de Amazon OpenSearch Service en la documentación de Amazon OpenSearch Service.

Identifique o genere credenciales de autenticación básica HTTP, *aosUser* y *aosPassword* para su dominio.

Para configurar una conexión a OpenSearch Service:

1. En AWS Secrets Manager, cree un secreto con sus credenciales de OpenSearch Service. Para crear un secreto en Secrets Manager, siga el tutorial disponible en Crear un secreto AWS Secrets Manager en la documentación AWS Secrets Manager. Después de crear el secreto, conserve el nombre secreto, *secretName*, para el siguiente paso.

   - Al seleccionar pares clave/valor, genere un par para la clave `opensearch.net.http.auth.user` con el valor *aosUser*.
   - Al seleccionar pares clave/valor, genere un par para la clave `opensearch.net.http.auth.pass` con el valor *aosPassword*.

2. En la consola de AWS Glue, genere una conexión mediante los pasos que se indican en the section called “Adición de una conexión de AWS Glue”. Tras crear la conexión, conserve el nombre de la conexión, *connectionName*, para el uso futuro en AWS Glue.

   - Al seleccionar un tipo de conexión, seleccione OpenSearch Service.
   - Al seleccionar un punto de conexión de dominio, proporcione *aosEndpoint*.
   - Al seleccionar un puerto, proporcione *aosPort*.
   - Al seleccionar un secreto AWS, proporcione un *secretName*.

Crear un nodo fuente de OpenSearch Service

Requisitos previos necesarios

- Una conexión AWS Glue de OpenSearch Service, configurada con un AWS Secrets Manager secreto, como se describe en la sección anterior, the section called “Crear una conexión con OpenSearch Service”.
- Permisos adecuados en el trabajo para leer el secreto utilizado por la conexión.
- Un índice de OpenSearch Service del que quiera leer, *aosIndex*.

Agregar un origen de datos de OpenSearch Service

Para añadir un nodo de origen de datos: OpenSearch Service:

1. Elija la conexión para el origen de datos de OpenSearch Service. Dado que lo ha creado, debería estar disponible en el menú desplegable. Si es necesario crear una conexión, elija Crear conexión de OpenSearch Service. Para obtener más información, consulte la sección the section called “Crear una conexión con OpenSearch Service” anterior.

   Una vez que haya elegido una conexión, puede ver las propiedades de la conexión mediante un clic en Ver propiedades.
2. Proporcione Index, el índice que desea leer.
3. Si lo desea, proporcione Query, una consulta de OpenSearch para obtener resultados más específicos. Para obtener más información sobre cómo escribir consultas de OpenSearch, consulte the section called “Lea de OpenSearch Service”.
4. En las propiedades personalizadas de OpenSearch Service, ingrese los parámetros y valores según sea necesario.

Crear un nodo de destino de OpenSearch Service

Requisitos previos necesarios

- Una conexión AWS Glue de OpenSearch Service, configurada con un AWS Secrets Manager secreto, como se describe en la sección anterior, the section called “Crear una conexión con OpenSearch Service”.
- Permisos adecuados en el trabajo para leer el secreto utilizado por la conexión.
- Un índice de OpenSearch Service al que desearía escribir, *`aosIndex`*.

Agregar un destino de datos de OpenSearch Service

Para añadir un nodo de Destino de datos: OpenSearch Service:

1. Elija la conexión para el origen de datos de OpenSearch Service. Dado que lo ha creado, debería estar disponible en el menú desplegable. Si es necesario crear una conexión, elija Crear conexión de OpenSearch Service. Para obtener más información, consulte la sección the section called “Crear una conexión con OpenSearch Service” anterior.

Una vez que haya elegido una conexión, puede ver las propiedades de la conexión mediante un clic en Ver propiedades.

2. Proporcione Index, el índice que desea leer.
3. En las propiedades personalizadas de OpenSearch Service, ingrese los parámetros y valores según sea necesario.

Opciones avanzadas

Puede brindar opciones avanzadas al crear un nodo de OpenSearch Service. Estas opciones son las mismas que las disponibles cuando se programa AWS Glue para scripts de Spark.

Consulte the section called “Conexiones de OpenSearch Service”.

Conectarse a Salesforce en AWS Glue Studio

Salesforce proporciona un software de gestión de relaciones con los clientes (CRM) que le ayuda con las ventas, el servicio al cliente, el comercio electrónico y mucho más. Si es usuario de Salesforce, puede conectarse AWS Glue a su cuenta de Salesforce. A continuación, puede utilizar Salesforce como fuente o destino de datos en sus trabajos de ETL. Ejecute estos trabajos para transferir datos entre Salesforce y AWS los servicios u otras aplicaciones compatibles.

Temas

- AWS Glue soporte para Salesforce
- Políticas que contienen las operaciones de la API para crear y usar conexiones
- Configuración de Salesforce
- Configuración de conexiones de Salesforce
- Lectura de entidades de Salesforce
- Escribir a Salesforce
- Opciones de conexión de Salesforce
- Limitaciones del conector de Salesforce
- Configurar el flujo OAuth del portador JWT para Salesforce

AWS Glue soporte para Salesforce

AWS Glue admite Salesforce de la siguiente manera:

¿Se admite como fuente?

Sí. Puede utilizar los trabajos de AWS Glue ETL para consultar datos de Salesforce.

¿Se admite como objetivo?

Sí. Puede usar AWS Glue ETL para escribir registros en Salesforce.

Versiones de la API de Salesforce compatibles

Se admiten las siguientes versiones de la API de Salesforce

- v58.0
- v5.9.0
- v6.0.0

Políticas que contienen las operaciones de la API para crear y usar conexiones

En el siguiente ejemplo de política se describen los permisos de AWS IAM necesarios para crear y utilizar conexiones. Si va a crear un nuevo rol, cree una política que contenga lo siguiente:

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:ListConnectionTypes",
                "glue:DescribeConnectionType",
                "glue:RefreshOAuth2Tokens"
                "glue:ListEntities",
                "glue:DescribeEntity"
            ],
            "Resource": "*"
        }
    ]
}
```

También puedes usar las siguientes políticas de IAM para permitir el acceso:

- AWSGlueServiceRole— Otorga acceso a los recursos que varios AWS Glue procesos necesitan para ejecutarse en su nombre. Estos recursos incluyen AWS Glue Amazon S3, IAM, CloudWatch

Logs y Amazon EC2. Si sigue la convención de nomenclatura de los recursos especificada en esta política, AWS Glue los procesos tienen los permisos necesarios. Normalmente, esta política se asocia a los roles que se especifican a la hora de definir rastreadores, trabajos y puntos de conexión de desarrollo.

- AWSGlueConsoleFullAccess— Otorga acceso total a AWS Glue los recursos cuando una identidad a la que está asociada la política utiliza la consola AWS de administración. Si sigue la convención de nomenclatura para los recursos especificados en esta política, los usuarios dispondrán de todas las funciones de la consola. Esta política suele estar asociada a los usuarios de la AWS Glue consola.

Configuración de Salesforce

Antes de poder utilizar Salesforce AWS Glue para transferir datos hacia o desde Salesforce, debe cumplir estos requisitos:

Requisitos mínimos

Los requisitos mínimos son los siguientes:

- Tiene una cuenta de Salesforce.
- Su cuenta de Salesforce está habilitada para el acceso a la API. El acceso a la API está habilitado de forma predeterminada para las ediciones Enterprise, Unlimited, Developer y Performance.
- Su cuenta de Salesforce le permite instalar aplicaciones conectadas. Si no tiene acceso a esta funcionalidad, póngase en contacto con su administrador de Salesforce. Para obtener más información, consulte Aplicaciones conectadas en la ayuda de Salesforce.

Si cumples estos requisitos, estás listo para conectarte AWS Glue a tu cuenta de Salesforce. AWS Glue gestiona los requisitos restantes con la aplicación conectada AWS gestionada.

La aplicación conectada AWS gestionada para Salesforce

La aplicación conectada AWS gestionada le ayuda a crear conexiones de Salesforce en menos pasos. En Salesforce, una aplicación conectada es un marco que autoriza a aplicaciones externas, por ejemplo AWS Glue, a acceder a sus datos de Salesforce.

- Cree una conexión con Salesforce mediante la consola. AWS Glue
- Al configurar la conexión, defina el tipo de concesión de OAuth como Código de autorización.

Configuración de conexiones de Salesforce

Para configurar una conexión de Salesforce:

1. En AWS Secrets Manager, crea un secreto con los siguientes detalles:
   a. Para el tipo de concesión JWT_TOKEN, el secreto debe contener la clave JWT_TOKEN con su valor.
   b. Para el tipo de AuthorizationCode concesión: en el caso de una aplicación conectada gestionada por el cliente, el secreto debe contener la aplicación conectada Consumer Secret con la clave. `USER_MANAGED_CLIENT_APPLICATION_CLIENT_SECRET` En el caso de una aplicación conectada AWS gestionada, un secreto vacío o un secreto con algún valor temporal.
   c. Nota: Debes crear un secreto por cada conexión en AWS Glue.
2. En el catálogo AWS Glue de datos, cree una conexión siguiendo los pasos que se indican a continuación:
   a. Al seleccionar un tipo de conexión, seleccione Salesforce.
   b. Proporcione la INSTANCE_URL de la Salesforce a la que desea conectarse.
   c. Proporcione el entorno de Salesforce.
   d. Seleccione la función de AWS IAM que AWS Glue pueda asumir y que tenga permisos para las siguientes acciones:
   e. Seleccione el tipo de concesión de OAuth2 que desee utilizar para las conexiones. El tipo de concesión determina cómo AWS Glue se comunica con Salesforce para solicitar el acceso a sus datos. Su elección afecta a los requisitos que debe cumplir antes de crear la conexión. Puede elegir uno de estos tipos:
      - Tipo de subvención JWT_BEARER: este tipo de subvención funciona bien en escenarios de automatización, ya que permite crear un token web JSON (JWT) por adelantado con los permisos de un usuario concreto en la instancia de Salesforce. El creador tiene el control sobre el tiempo de validez del JWT. AWS Glue puede usar el JWT para obtener un token de acceso que se utiliza para llamar a las API de Salesforce.

        Este flujo requiere que el usuario haya creado una aplicación conectada en su instancia de Salesforce que permita emitir tokens de acceso basados en JWT para los usuarios.

        Para obtener información sobre cómo crear una aplicación conectada para el flujo OAuth del portador de JWT, consulte el flujo del portador JWT de OAuth 2.0 para la integración. server-to-server Para configurar el flujo portador de JWT con la aplicación conectada a Salesforce, consulte. Configurar el flujo OAuth del portador JWT para Salesforce

- Tipo de concesión AUTHORIZATION_CODE: este tipo de concesión se considera un OAuth «de tres vías», ya que se basa en redirigir a los usuarios al servidor de autorización externo para autenticar al usuario. Se utiliza para crear conexiones a través de la consola. AWS Glue El usuario que crea una conexión puede utilizar de forma predeterminada una aplicación AWS Glue conectada (aplicación cliente AWS Glue gestionada), en la que no necesita proporcionar ninguna información relacionada con OAuth, excepto la URL de su instancia de Salesforce. La AWS Glue consola redirigirá al usuario a Salesforce, donde deberá iniciar sesión y permitir AWS Glue que los permisos solicitados accedan a su instancia de Salesforce.

  Los usuarios aún pueden optar por crear su propia aplicación conectada en Salesforce y proporcionar su propio ID y secreto de cliente al crear conexiones a través de la consola. AWS Glue En este escenario, seguirán siendo redirigidos a Salesforce para iniciar sesión y autorizar el acceso AWS Glue a sus recursos.

  Este tipo de concesión da como resultado un token de actualización y un token de acceso. El token de acceso es de corta duración y se puede actualizar automáticamente sin la interacción del usuario mediante el token de actualización.

  Para obtener información sobre cómo crear una aplicación conectada para el flujo de código de autorización OAuth, consulte Configurar una aplicación conectada para el flujo de códigos de autorización y credenciales.

f. Seleccione el `secretName` que desee usar para esta conexión AWS Glue para colocar los tokens.

g. Seleccione las opciones de red si quiere usar su red. Conceda permiso de lectura `secretName` al rol de IAM asociado a su AWS Glue trabajo.

3. Conceda permiso de lectura al rol de IAM asociado a su AWS Glue trabajo. `secretName`

4. En la configuración de su AWS Glue trabajo, `connectionName` indíquelo como conexión de red adicional.

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:DescribeSecret",
                "secretsmanager:GetSecretValue",
```

```
                "secretsmanager:PutSecretValue",
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterface",
                "ec2:DeleteNetworkInterface",
            ],
            "Resource": "*"
        }
    ]
}
```

Lectura de entidades de Salesforce

Requisito previo

Un objeto de Salesforce del que le gustaría leer. Necesitará el nombre del objeto, como `Account` o o`Case`. `Opportunity`

Ejemplo:

```
salesforce_read = glueContext.create_dynamic_frame.from_options(
    connection_type="salesforce",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "Account",
        "API_VERSION": "v60.0"
    }
```

Consultas de particionamiento

Puedes proporcionar las opciones adicionales de Spark `PARTITION_FIELD` `LOWER_BOUNDUPPER_BOUND`, y `NUM_PARTITIONS` si quieres utilizar la simultaneidad en Spark. Con estos parámetros, la consulta original se dividiría en un `NUM_PARTITIONS` número de subconsultas que las tareas de Spark pueden ejecutar simultáneamente.

- `PARTITION_FIELD`: el nombre del campo que se utilizará para particionar la consulta.
- `LOWER_BOUND`: un valor límite inferior inclusivo del campo de partición elegido.

  Para el campo de marca de tiempo, aceptamos el formato de marca de tiempo de Spark utilizado en las consultas SQL de Spark.

  Ejemplos de valores válidos:

```
"TIMESTAMP \"1707256978123\""
"TIMESTAMP ‘2024-02-06 22:02:58.123 UTC'"
"TIMESTAMP \"2018-08-08 00:00:00 Pacific/Tahiti\"
"TIMESTAMP \"2018-08-08 00:00:00\""
"TIMESTAMP \"-123456789\" Pacific/Tahiti"
"TIMESTAMP \"1702600882\""
```

- UPPER_BOUND: un valor límite superior exclusivo del campo de partición elegido.
- NUM_PARTITIONS: el número de particiones.

Ejemplo:

```
salesforce_read = glueContext.create_dynamic_frame.from_options(
    connection_type="salesforce",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "Account",
        "API_VERSION": "v60.0",
        "PARTITION_FIELD": "SystemModstamp"
        "LOWER_BOUND": "TIMESTAMP '2021-01-01 00:00:00 Pacific/Tahiti'"
        "UPPER_BOUND": "TIMESTAMP '2023-01-10 00:00:00 Pacific/Tahiti'"
        "NUM_PARTITIONS": "10"
    }
```

Escribir a Salesforce

Requisitos previos

Un objeto de Salesforce al que le gustaría escribirle. Necesitará el nombre del objeto, como `Account` o o`Case`. `Opportunity`

El conector de Salesforce admite cuatro operaciones de escritura:

- INSERT
- UPSERT
- UPDATE
- DELETE

Al utilizar la operación de UPSERT escritura, se ID_FIELD_NAMES debe proporcionar para especificar el campo de ID externo de los registros.

Ejemplo

```
salesforce_write = glueContext.write_dynamic_frame.from_options(
    frame=frameToWrite,
    connection_type="salesforce",
    connection_options={
        "connectionName": "connectionName",
        "ENTITY_NAME": "Account",
        "API_VERSION": "v60.0",
        "WRITE_OPERATION": "INSERT"
    }
```

Opciones de conexión de Salesforce

Las siguientes son opciones de conexión para Salesforce:

- ENTITY_NAME(Cadena): (obligatorio) Se utiliza para lectura/escritura. El nombre de su objeto en Salesforce.
- API_VERSION(Cadena): (obligatorio) Se usa para lectura/escritura. Versión de la API Rest de Salesforce que desea utilizar.
- SELECTED_FIELDS(Lista<String>) - Predeterminada: vacía (SELECT *). Se usa para leer. Columnas que desee seleccionar para el objeto.
- FILTER_PREDICATE(Cadena) - Predeterminado: vacío. Se usa para leer. Debe estar en el formato Spark SQL.
- QUERY(String) - Predeterminado: vacío. Se usa para leer. Consulta SQL completa de Spark.
- PARTITION_FIELD(String): se usa para leer. Campo que se utilizará para particionar la consulta.
- LOWER_BOUND(Cadena): se usa para leer. Un valor de límite inferior inclusivo del campo de partición elegido.
- UPPER_BOUND(Cadena): se usa para leer. Un valor límite superior exclusivo del campo de partición elegido.
- NUM_PARTITIONS(Entero): predeterminado: 1. Se usa para leer. Número de particiones para leer.
- IMPORT_DELETED_RECORDS(Cadena) - Predeterminado: FALSO. Se usa para leer. Para obtener los registros eliminados durante la consulta.

- `WRITE_OPERATION`(String) - Predeterminado: INSERT. Se usa para escribir. El valor debe ser INSERT, UPDATE, UPSERT, DELETE.
- `ID_FIELD_NAMES`(String) - Predeterminado: nulo. Se usa solo para UPSERT.

Limitaciones del conector de Salesforce

Las siguientes son limitaciones del conector de Salesforce:

- Solo admitimos Spark SQL y Salesforce SOQL no.
- No se admiten los marcadores de trabajo.

Configurar el flujo OAuth del portador JWT para Salesforce

Consulte la documentación pública de Salesforce para habilitar la server-to-server integración con los JSON Web Tokens de OAuth 2.0.

Creación de un par de archivos PEM de certificados y claves

Cree un par de archivos PEM de certificados y claves

```
openssl req -newkey rsa:4096 -new -nodes -x509 -days 3650 -keyout key.pem -out cert.pem
```

Creación de una aplicación conectada a Salesforce con JWT

1. Inicie sesión en Salesforce, haga clic en el engranaje de configuración situado en la parte superior derecha y seleccione Configuración.
2. A la izquierda, navega hasta App Manager. (Herramientas de plataforma > Aplicaciones > Administrador de aplicaciones)
3. Seleccione Nueva aplicación de conexión.
4. Proporcione un nombre para la aplicación y deje que el resto se complete automáticamente.
5. Marca la casilla Habilitar la configuración de OAuth.
6. Establece una URL de devolución de llamada. No se usará para JWT, así que puedes usar https://localhost.
7. Marca la casilla Usar firmas digitales.
8. Cargue el archivo cert.pem creado anteriormente.

9. Añada los permisos necesarios:
   a. Gestione los datos de los usuarios a través de las API (api).
   b. Acceda a permisos personalizados (custom_permissions).
   c. Acceda al servicio de URL de identidad (id, perfil, correo electrónico, dirección, teléfono).
   d. Acceda a los identificadores de usuario únicos (openid).
   e. Realice solicitudes en cualquier momento (refresh_token, offline_access).
10. Marque la casilla Emitir tokens de acceso basados en el token web JSON (JWT) para los usuarios designados.
11. Seleccione Guardar.
12. Elija Continuar.
13. Elija Manage Consumer Details.
14. Copia la clave de consumidor (ID de cliente).
15. Copie el secreto del consumidor (secreto del cliente).
16. Haga clic en Cancel (Cancelar).

Generar un token web JSON (JWT)

1. Convierte el key pair en pkcs12 (establece una contraseña de exportación cuando se te pida).

```
openssl pkcs12 -export -in cert.pem -inkey key.pem -name jwtcert > jwtcert.p12
```

2. Cree un almacén de claves Java a partir de pkcs12 (establezca una contraseña del almacén de claves de destino cuando se le solicite y proporcione la contraseña de exportación anterior para la contraseña del almacén de claves de origen).

```
keytool -importkeystore -srckeystore jwtcert.p12 -destkeystore keystore.jks -
srcstoretype pkcs12 -alias jwtcert
```

3. Confirme que keystore.jks incluya el alias jwtcert (introduzca la contraseña anterior del almacén de claves de destino cuando se le pida).

```
keytool -keystore keystore.jks -list
```

4. Utilice la clase Java JWTExample que se proporciona en la documentación de Salesforce para generar el token firmado.
   a. Edite los valores en ClaimArray según sea necesario:

- ClaimArray [0] = identificador de cliente
- ClaimArray [1] = ID de usuario de Salesforce
- ClaimArray [2] = URL de inicio de sesión de Salesforce
- ClaimArray [4] = fecha de caducidad en milisegundos desde la época. 3660624000000 es 2085-12-31.

b. Sustituya path/to/keystore por la ruta correcta a su keystore.jks.

c. Sustituya keystore/password por la contraseña del almacén de claves de destino que introdujiste

d. Sustituya privatekeypassword por la contraseña del almacén de claves de origen que ingresó

e. Compile el código. El código depende del códec Apache Commons para la codificación base64.

```
javac -classpath ".:./commons-codec-1.16.1.jar" JWTExample.java
```

f. Ejecute el código.

```
java -classpath ".:commons-codec-1.16.1.jar" JWTExample
```

5. Una vez que se hayan creado la aplicación conectada y el JWT, el usuario aún debe estar autorizado para utilizar la aplicación. Consulte el paso 3 en https://mannharleen.github.io/2020-03-03-salesforce-jwt/ para ver dos enfoques.

Una vez completados los pasos anteriores, se generará un token web JSON (JWT) que se puede utilizar para obtener los tokens de acceso de Salesforce.

Ejemplo de entrada:

```
export password for pkcs12: awsglue
destination keystore password for jks: awsglue
source keystore password for jks: awsglue

claimArray[0] = “client-id”;
claimArray[1] = “my@email.com”;
claimArray[2] = "https://login.salesforce.com“;
claimArray[3] = "3660624000000";

path to keystore: ./keystore.jks
keystore password: awsglue
privatekey password: awsglue
```

Ejemplo de salida:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0Ij
```

Enlaces útiles:

- https://www.base64encode.org/
- https://jwt.io/
- https://help.salesforce.com/s/articleView?id=sf.remoteaccess_oauth_jwt_flow.htm

JWTExample.java:

```
import org.apache.commons.codec.binary.Base64;
import java.io.*;
import java.security.*;
import java.text.MessageFormat;

public class JWTExample {

  public static void main(String[] args) {

    String header = "{\"alg\":\"RS256\"}";
    String claimTemplate = "'{'\"iss\": \"{0}\", \"sub\": \"{1}\", \"aud\": \"{2}\",
 \"exp\": \"{3}\"'}'";

    try {
      StringBuffer token = new StringBuffer();

      //Encode the JWT Header and add it to our string to sign
      token.append(Base64.encodeBase64URLSafeString(header.getBytes("UTF-8")));

      //Separate with a period
      token.append(".");

      //Create the JWT Claims Object
      String[] claimArray = new String[5];
      claimArray[0] = "value";
      claimArray[1] = "my@email.com";
      claimArray[2] = "https://login.salesforce.com";
      claimArray[3] = Long.toString( ( System.currentTimeMillis()/1000 ) + 300);
      MessageFormat claims;
```

```
      claims = new MessageFormat(claimTemplate);
      String payload = claims.format(claimArray);

      //Add the encoded claims object
      token.append(Base64.encodeBase64URLSafeString(payload.getBytes("UTF-8")));

      //Load the private key from a keystore
      KeyStore keystore = KeyStore.getInstance("JKS");
      keystore.load(new FileInputStream("./keystore.jks"), "awsglue".toCharArray());
      PrivateKey privateKey = (PrivateKey) keystore.getKey("jwtcert",
 "awsglue".toCharArray());

      //Sign the JWT Header + "." + JWT Claims Object
      Signature signature = Signature.getInstance("SHA256withRSA");
      signature.initSign(privateKey);
      signature.update(token.toString().getBytes("UTF-8"));
      String signedPayload = Base64.encodeBase64URLSafeString(signature.sign());

      //Separate with a period
      token.append(".");

      //Add the encoded signature
      token.append(signedPayload);

      System.out.println(token.toString());

    } catch (Exception e) {
        e.printStackTrace();
    }
  }
}
```

Conexión a SAP HANA en AWS Glue Studio

AWS Glue proporciona soporte integrado para SAP HANA. AWS Glue Studio proporciona una interfaz visual para conectarse a SAP HANA, crear trabajos de integración de datos y ejecutarlos en el tiempo de ejecución de Spark sin servidor de AWS Glue Studio.

Temas

- Crear una conexión SAP HANA
- Crear un nodo de origen de SAP HANA
- Crear un nodo de destino de SAP HANA

- Opciones avanzadas

Crear una conexión SAP HANA

Para conectarse a SAP HANA desde AWS Glue, tendrá que crear y almacenar sus credenciales de SAP HANA en un AWS Secrets Manager secreto y, a continuación, asociar ese secreto a una conexión de AWS Glue de SAP HANA. Deberá configurar la conectividad de red entre su servicio SAP HANA y AWS Glue.

Requisitos previos:

- Si su servicio SAP HANA está en una VPC de Amazon, configure Amazon VPC para permitir que su trabajo de AWS Glue se comunique con el servicio SAP HANA sin que el tráfico atraviese la Internet pública.

  En Amazon VPC, identifique o cree una VPC, una subred y un grupo de seguridad que AWS Glue utilizará al ejecutar el trabajo. Además, debe asegurarse de que Amazon VPC esté configurada para permitir el tráfico de red entre su punto de conexión SAP HANA y esta ubicación. Su trabajo deberá establecer una conexión TCP con su puerto JDBC de SAP HANA. Para obtener más información sobre los puertos de SAP HANA, consulte la documentación de SAP HANA. Según el diseño de la red, esto puede requerir cambios en las reglas de los grupos de seguridad, las ACL de red, las puertas de enlace de NAT y las conexiones entre pares.

Para configurar una conexión a SAP HANA:

1. En AWS Secrets Manager, cree un secreto con sus credenciales de SAP HANA. Para crear un secreto en Secrets Manager, siga el tutorial disponible en Crear un secreto AWS Secrets Manager en la documentación AWS Secrets Manager. Después de crear el secreto, conserve el nombre secreto, *`secretName`*, para el siguiente paso.

   - Al seleccionar pares clave/valor, genere un par para la clave `user` con el valor *`saphanaUsername`*.
   - Al seleccionar pares clave/valor, genere un par para la clave `password` con el valor *`saphanaPassword`*.
2. En la consola de AWS Glue, genere una conexión mediante los pasos que se indican en the section called “Adición de una conexión de AWS Glue”. Tras crear la conexión, conserve el nombre de la conexión, *`connectionName`*, para el uso futuro en AWS Glue.

- Al seleccionar un tipo de conexión, seleccione SAP HANA.
- Al proporcionar la URL de SAP HANA, proporcione la URL de su instancia.

  Las URL de JDBC de SAP HANA tienen el formato `jdbc:sap://saphanaHostname:saphanaPort/?databaseName=saphanaDBname,Paramete`

  AWS Glue requiere los siguientes parámetros de URL de JDBC:

  - `databaseName` — Una base de datos predeterminada en SAP HANA a la que conectarse.
- Al seleccionar un secreto AWS, proporcione un *`secretName`*.

Tras crear una conexión con AWS Glue SAP HANA, deberá realizar los siguientes pasos antes de ejecutar su trabajo de AWS Glue:

- Otorga permiso al rol de IAM asociado al trabajo de AWS Glue para leer el *`secretName`*.

Crear un nodo de origen de SAP HANA

Requisitos previos necesarios

- Una conexión AWS Glue SAP HANA, configurada con un AWS Secrets Manager secreto, tal y como se describe en la sección anterior, the section called “Crear una conexión SAP HANA”.
- Permisos adecuados en el trabajo para leer el secreto utilizado por la conexión.
- Una tabla de SAP HANA de la que le gustaría leer, *`tableName`*, o consultar *`targetQuery`*.

  Se puede especificar una tabla con un nombre de tabla y un nombre de esquema de SAP HANA, en la forma *`schemaName`*`.`*`tableName`*. El nombre del esquema y el separador "." no son necesarios si la tabla está en el esquema predeterminado, "public". Esto se llama *`tableIdentifier`*. Tenga en cuenta que la base de datos se proporciona como un parámetro de URL de JDBC en `connectionName`.

Agregar un origen de datos de SAP HANA

Para agregar un nodo de Origen de datos: SAP HANA:

1. Elija la conexión para el origen de datos de SAP HANA. Dado que lo ha creado, debería estar disponible en el menú desplegable. Si es necesario crear una conexión, elija Crear conexión de

SAP HANA. Para obtener más información, consulte la sección the section called “Crear una conexión SAP HANA” anterior.

Una vez que haya elegido una conexión, puede ver las propiedades de la conexión mediante un clic en Ver propiedades.

2. Elija una opción de origen de SAP HANA:
   - Elija una sola tabla: acceda a todos los datos de una sola tabla.
   - Ingresar una consulta personalizada: permite acceder a un conjunto de datos de varias tablas en función del consulta personalizada.
3. Si eligió una sola tabla, ingrese *`tableName`*.

   Si eligió Introducir una consulta personalizada, introduzca una consulta SQL SELECT.
4. En las propiedades personalizadas de SAP HANA, ingrese los parámetros y valores según sea necesario.

Crear un nodo de destino de SAP HANA

Requisitos previos necesarios

- Una conexión AWS Glue SAP HANA, configurada con un AWS Secrets Manager secreto, como se describe en la sección anterior, the section called “Crear una conexión SAP HANA”.
- Permisos adecuados en el trabajo para leer el secreto utilizado por la conexión.
- Una tabla de SAP HANA en la que le gustaría escribir, *`tableName`*.

  Se puede especificar una tabla con un nombre de tabla y un nombre de esquema de SAP HANA, en la forma *`schemaName`*`.`*`tableName`*. El nombre del esquema y el separador "." no son necesarios si la tabla está en el esquema predeterminado, "public". Esto se llama *`tableIdentifier`*. Tenga en cuenta que la base de datos se proporciona como un parámetro de URL de JDBC en `connectionName`.

Agregar un destino de datos de SAP HANA

Para añadir un nodo de destino de datos: SAP HANA:

1. Elija la conexión para el origen de datos de SAP HANA. Dado que lo ha creado, debería estar disponible en el menú desplegable. Si es necesario crear una conexión, elija Crear conexión de

SAP HANA. Para obtener más información, consulte la sección the section called “Crear una conexión SAP HANA” anterior.

Una vez que haya elegido una conexión, puede ver las propiedades de la conexión mediante un clic en Ver propiedades.

2. Configure el nombre de la tabla proporcionando *`tableName`*.
3. En las propiedades personalizadas de Teradata, ingrese los parámetros y valores según sea necesario.

Opciones avanzadas

Puede brindar opciones avanzadas al crear un nodo de SAP HANA. Estas opciones son las mismas que las disponibles cuando se programa AWS Glue para scripts de Spark.

Consulte the section called “Conexiones SAP HANA”.

Conectarse a Snowflake en AWS Glue Studio

> **Note**
>
> Puede usar AWS Glue para Spark para leer y escribir en tablas de Snowflake en AWS Glue 4.0 y versiones posteriores. Para configurar una conexión de Snowflake con trabajos de AWS Glue mediante programación, consulte Conexiones Redshift.

AWS Glue proporciona soporte integrado para Snowflake. AWS Glue Studio proporciona una interfaz visual para conectarse a Snowflake, crear trabajos de integración de datos y ejecutarlos en el tiempo de ejecución de Spark sin servidor de AWS Glue Studio.

Temas

- Crear una conexión con Snowflake
- Creación de un nodo de origen de Snowflake
- Creación de un nodo de destino de Snowflake
- Opciones avanzadas

Crear una conexión con Snowflake

Al agregar un origen de datos: nodo Snowflake en AWS Glue Studio, puede elegir una conexión de Snowflake en AWS Glue existente o crear una nueva conexión. Debe elegir un tipo de conexión `SNOWFLAKE` y no un tipo de conexión `JDBC` configurado para conectarse a Snowflake. Utilice el siguiente procedimiento para crear una conexión con Snowflake en AWS Glue:

Crear una conexión con Snowflake

1. En Snowflake, genere un usuario *`snowflakeUser`* y una contraseña *`snowflakePassword`*.
2. Determine con qué almacén de Snowflake interactuará este usuario, *`snowflakeWarehouse`*. Configúrelo como `DEFAULT_WAREHOUSE` para *`snowflakeUser`* en Snowflake o recuérdelo para el siguiente paso.
3. En AWS Secrets Manager, cree un secreto con sus credenciales de Snowflake. Para crear un secreto en Secrets Manager, siga el tutorial disponible en Crear un secreto AWS Secrets Manager en la documentación AWS Secrets Manager. Después de crear el secreto, conserve el nombre secreto, *`secretName`*, para el siguiente paso.
   - Al seleccionar pares clave/valor, cree un par para *`snowflakeUser`* con la clave `sfUser`.
   - Al seleccionar pares clave/valor, cree un par para *`snowflakePassword`* con la clave `sfPassword`.
   - *Al seleccionar* ***pares clave/valor****, cree un par para snowflakeWarehouse* con la clave `sfWarehouse`. Esto no es necesario si hay un valor predeterminado en Snowflake.
4. En el catálogo de datos de AWS Glue, cree una conexión mediante los pasos que se indican en Agregar una conexión en AWS Glue. Tras crear la conexión, conserve el nombre de la conexión, *`connectionName`*, para el siguiente paso.
   - Al seleccionar un tipo de conexión, seleccione Snowflake.
   - Al seleccionar la URL de Snowflake, proporcione el nombre de host de la instancia de Snowflake. La URL utilizará un nombre de host en el formulario *`account_identifier`*`.snowflakecomputing.com`.
   - Al seleccionar un secreto AWS, proporcione un *`secretName`*.

Creación de un nodo de origen de Snowflake

Permisos necesarios

los trabajos de AWS Glue Studio que utilizan orígenes de datos de Snowflake requieren permisos adicionales. Para obtener más información sobre cómo agregar permisos a los trabajos de ETL, consulte Revisar los permisos de IAM necesarios para los trabajos de ETL.

las conexiones de `SNOWFLAKE` en AWS Glue utilizan un secreto de AWS Secrets Manager para proporcionar información sobre las credenciales. Sus roles de vista previa de datos y trabajos en AWS Glue Studio deben tener permiso para leer este secreto.

Agregar un origen de datos de Snowflake

Requisitos previos:

- Un secreto de AWS Secrets Manager para sus credenciales de Snowflake
- Una conexión del catálogo de datos de AWS Glue tipo Snowflake

Para agregar un nodo de Origen de datos: Snowflake:

1. Elija la conexión para el origen de datos de Snowflake. Esto supone que la conexión ya existe y que puede seleccionar entre las conexiones existentes. Si necesita crear una conexión, elija Crear conexión de Snowflake. Para más información, consulte Información general sobre el uso de conectores y conexiones.

   Una vez que haya elegido una conexión, puede ver las propiedades de la conexión mediante un clic en Ver propiedades. La información sobre la conexión está visible, como la URL, los grupos de seguridad, la subred, la zona de disponibilidad, la descripción y las marcas horarias creadas (UTC) y actualizadas por última vez (UTC).
2. Elija una opción de origen de Snowflake:

   - Elija una sola tabla: esta es la tabla que contiene los datos a los que desea acceder desde una sola tabla de Snowflake.
   - Ingresar una consulta personalizada: permite acceder a un conjunto de datos de varias tablas de Snowflake en función de la consulta personalizada.
3. Si ha elegido una sola tabla, ingrese el nombre de un esquema de Snowflake.

O bien, elija Ingresar consulta personalizada. Elija esta opción para acceder a un conjunto de datos personalizado desde varias tablas de Snowflake. Al elegir esta opción, ingrese la consulta Snowflake.

4. En las opciones de rendimiento y seguridad (opcional),

   - Habilite la función desplegable de consultas: elija si quiere transferir el trabajo a la instancia de Snowflake.

5. En las propiedades personalizadas de Snowflake (opcional), ingrese los parámetros y valores según sea necesario.

Creación de un nodo de destino de Snowflake

Permisos necesarios

los trabajos de AWS Glue Studio que utilizan orígenes de datos de Snowflake requieren permisos adicionales. Para obtener más información sobre cómo agregar permisos a los trabajos de ETL, consulte Revisar los permisos de IAM necesarios para los trabajos de ETL.

las conexiones de `SNOWFLAKE` en AWS Glue utilizan un secreto de AWS Secrets Manager para proporcionar información sobre las credenciales. Sus roles de vista previa de datos y trabajos en AWS Glue Studio deben tener permiso para leer este secreto.

Agregar un destino de datos de Snowflake

Crear de un nodo de destino de Snowflake:

1. Elija una tabla Snowflake existente como destino o ingrese un nombre de tabla nuevo.
2. Al utilizar el nodo destino de datos: Snowflake, puede elegir entre las siguientes opciones:

   - APPEND: si ya existe una tabla, coloque todos los datos nuevos en la tabla como un inserto. Si la tabla no existe, créela y, a continuación, inserte todos los datos nuevos.
   - MERGE: AWS Glue actualizará o anexará datos a la tabla de destino en función de las condiciones que especifique.

     Elija opciones:

     - Elegir claves y acciones sencillas: elija las columnas que se usarán como claves de coincidencia entre los orígenes de datos y el conjunto de destinos de datos.

Especifique las siguientes opciones cuando coincidan:

- Actualice el registro del conjunto de datos de destino con los datos de origen.
- Elimine el registro del conjunto de datos de destino.

Especifique las siguientes opciones cuando no coincidan:

- Inserte los datos de origen como una nueva fila en el conjunto de datos de destino.
- No hacer nada.

- Ingrese una instrucción MERGE personalizada: a continuación, puede elegir Validar la instrucción MERGE para comprobar si la instrucción es válida o no.

- TRUNCATE: si ya existe una tabla, trunque los datos de la tabla al borrar primero el contenido de la tabla de destino. Si el truncado se realiza correctamente, inserte todos los datos. Si la tabla no existe, créela y, a continuación, inserte todos los datos. Si el truncado no es exitoso, la operación producirá un error.
- DROP: si una tabla ya existe, elimine los metadatos y los datos de la tabla. Si el borrado se realiza correctamente, inserte todos los datos. Si la tabla no existe, créela y, a continuación, inserte todos los datos. Si el descarte no es exitoso, la operación producirá un error.

Opciones avanzadas

Consulte las conexiones de Snowflake en la guía para desarrolladores de AWS Glue.

Conexión a Teradata Vantage en AWS Glue Studio

AWS Glue proporciona soporte integrado para Teradata Vantage. AWS Glue Studio proporciona una interfaz visual para conectarse a Teradata, crear trabajos de integración de datos y ejecutarlos en el tiempo de ejecución de Spark sin servidor de AWS Glue Studio.

Temas

- Crear una conexión Teradata Vantage
- Crear un nodo de origen de Teradata
- Crear un nodo de destino de Teradata
- Opciones avanzadas

Crear una conexión Teradata Vantage

Para conectarse a Teradata Vantage desde AWS Glue, tendrá que crear y almacenar sus credenciales de Teradata en un AWS Secrets Manager secreto y, a continuación, asociar ese secreto a una conexión de Teradata de AWS Glue.

Requisitos previos:

- Si accede a su entorno de Teradata a través de Amazon VPC, configure Amazon VPC para permitir que su trabajo de AWS Glue se comunique con el entorno de Teradata. No recomendamos acceder al entorno de Teradata a través de la Internet pública.

  En Amazon VPC, identifique o cree una VPC, una subred y un grupo de seguridad que AWS Glue utilizará al ejecutar el trabajo. Además, debe asegurarse de que Amazon VPC esté configurada para permitir el tráfico de red entre su instancia de Teradata y esta ubicación. Su trabajo deberá establecer una conexión TCP con su puerto de cliente de Teradata. Para obtener más información sobre los puertos de Teradata, consulte la documentación de Teradata.

  Según el diseño de la red, la conectividad segura de la VPC puede requerir cambios en Amazon VPC y otros servicios de red. Para obtener más información sobre la conectividad de AWS, consulte las Opciones de conectividad de AWS en la documentación de Teradata.

Para configurar una conexión de AWS Glue Teradata:

1. En la configuración de Teradata, identifique o cree un usuario y la contraseña con la que AWS Glue se conectará, *`teradataUser`* y *`teradataPassword`*. Para obtener más información, consulte la Información general de seguridad de Vantage en la documentación de Teradata.
2. En AWS Secrets Manager, cree un secreto con sus credenciales de Teradata. Para crear un secreto en Secrets Manager, siga el tutorial disponible en Crear un secreto AWS Secrets Manager en la documentación AWS Secrets Manager. Después de crear el secreto, conserve el nombre secreto, *`secretName`*, para el siguiente paso.

   - Al seleccionar pares clave/valor, genere un par para la clave `user` con el valor *`teradataUsername`*.
   - Al seleccionar pares clave/valor, genere un par para la clave `password` con el valor *`teradataPassword`*.

3. En la consola de AWS Glue, genere una conexión mediante los pasos que se indican en the section called “Adición de una conexión de AWS Glue”. Tras crear la conexión, conserve el nombre de la conexión, *connectionName*, para el siguiente paso.

   - Al seleccionar un tipo de conexión, seleccione Teradata.
   - Al proporcionar la URL de JDBC, proporcione la URL de su instancia. También puede codificar determinados parámetros de conexión separados por comas en la URL de JDBC. La URL debe tener el siguiente formato: `jdbc:teradata://`*teradataHostname*`/`*ParameterName*`=`*ParameterValue*`,`*ParameterName*

     Los parámetros de URL admitidos incluyen:

     - `DATABASE`— nombre de la base de datos del host a la que se accede de forma predeterminada.
     - `DBS_PORT`— el puerto de la base de datos, que se utiliza cuando se ejecuta en un puerto no estándar.
   - Al seleccionar un Tipo de credencial, seleccione AWS Secrets Manager y, a continuación, establezca AWS Secret en *secretName*.

4. En las siguientes situaciones, es posible que necesite una configuración adicional:

   - Para las instancias de Teradata alojadas AWS en una VPC de Amazon
     - Deberá proporcionar la información de conexión de Amazon VPC a la conexión de AWS Glue que define sus credenciales de seguridad de Teradata. Al crear o actualizar la conexión, configure los VPC, Subred y los grupos de seguridad en Opciones de red.

Crear un nodo de origen de Teradata

Requisitos previos necesarios

- Una conexión AWS Glue Teradata Vantage, configurada con un AWS Secrets Manager secreto, como se describe en la sección anterior, the section called “Crear una conexión Teradata Vantage”.
- Permisos adecuados en el trabajo para leer el secreto utilizado por la conexión.
- Una tabla de Teradata de la que desee leer, *tableName* o consultar *targetQuery*.

Agregar un origen de datos de Teradata

Para agregar un nodo de Origen de datos: Teradata:

1. Elija la conexión para el origen de datos de Teradata. Dado que lo ha creado, debería estar disponible en el menú desplegable. Si es necesario crear una conexión, elija Crear una nueva conexión. Para obtener más información, consulte la sección the section called “Crear una conexión Teradata Vantage” anterior.

   Una vez que haya elegido una conexión, puede ver las propiedades de la conexión mediante un clic en Ver propiedades.
2. Elija una opción de origen de Teradata:

   - Elija una sola tabla: acceda a todos los datos de una sola tabla.
   - Ingresar una consulta personalizada: permite acceder a un conjunto de datos de varias tablas en función del consulta personalizada.
3. Si eligió una sola tabla, ingrese *`tableName`*.

   Si eligió Introducir una consulta personalizada, introduzca una consulta SQL SELECT.
4. En las propiedades personalizadas de Teradata, ingrese los parámetros y valores según sea necesario.

Crear un nodo de destino de Teradata

Requisitos previos necesarios

- Una conexión AWS Glue Teradata Vantage, configurada con un AWS Secrets Manager secreto, como se describe en la sección anterior, the section called “Crear una conexión Teradata Vantage”.
- Permisos adecuados en el trabajo para leer el secreto utilizado por la conexión.
- Una tabla de Teradata a la que desearía escribir, *`tableName`*.

Agregar un destino de datos de Teradata

Para añadir un destino de datos: nodo de Teradata:

1. Elija la conexión para el origen de datos de Teradata. Dado que lo ha creado, debería estar disponible en el menú desplegable. Si es necesario crear una conexión, elija Crear conexión

de Teradata. Para más información, consulte Información general sobre el uso de conectores y conexiones.

Una vez que haya elegido una conexión, puede ver las propiedades de la conexión mediante un clic en Ver propiedades.

2. Configure el nombre de la tabla proporcionando *tableName*.
3. En las propiedades personalizadas de Teradata, ingrese los parámetros y valores según sea necesario.

Opciones avanzadas

Puede brindar opciones avanzadas al crear un nodo de Teradata. Estas opciones son las mismas que las disponibles cuando se programa AWS Glue para scripts de Spark.

Consulte the section called “Conexiones Teradata Vantage”.

Conexión a Vertica en AWS Glue Studio

AWS Glue proporciona soporte integrado para Vertica. AWS Glue Studio proporciona una interfaz visual para conectarse a Vertica, crear trabajos de integración de datos y ejecutarlos en el tiempo de ejecución de Spark sin servidor de AWS Glue Studio.

Temas

- Crear una conexión Vertica
- Crear un nodo de origen de Vertica
- Crear un nodo de destino de Vertica
- Opciones avanzadas

Crear una conexión Vertica

Requisitos previos:

- Un bucket o una carpeta de Amazon S3 para utilizar como almacenamiento temporario al leer y escribir en la base de datos, al que hace referencia *tempS3Path*.

> **Note**
>
> Al utilizar Vertica en las vistas previas de los datos de los trabajos de AWS Glue, puede que los archivos temporales no se eliminen automáticamente de *tempS3Path*. Para garantizar la eliminación de los archivos temporales, finalice directamente la sesión de vista previa de datos al seleccionar Finalizar sesión en el panel Vista previa de datos. Si no puede garantizar que la sesión de vista previa de datos finalice directamente, considere configurar Amazon S3 Lifecycle para eliminar los datos antiguos. Recomendamos eliminar los datos de más de 49 horas, en función del tiempo máximo de ejecución del trabajo más un margen. Para obtener más información sobre la configuración de Amazon S3 Lifecycle, consulte Administración del ciclo de vida del almacenamiento en la documentación de Amazon S3.

- Una política de IAM con los permisos adecuados para su ruta de Amazon S3 que pueda asociar a su puesto de trabajo de AWS Glue.
- Si su instancia de Vertica está en una VPC de Amazon, configure Amazon VPC para permitir que su trabajo de AWS Glue se comunique con la instancia de Vertica sin que el tráfico atraviese la Internet pública.

  En Amazon VPC, identifique o cree una VPC, una subred y un grupo de seguridad que AWS Glue utilizará al ejecutar el trabajo. Además, debe asegurarse de que Amazon VPC esté configurada para permitir el tráfico de red entre su instancia de Vertica y esta ubicación. Su trabajo tendrá que establecer una conexión TCP con el puerto de cliente de Vertica (el valor predeterminado es 5433). Según el diseño de la red, esto puede requerir cambios en las reglas de los grupos de seguridad, las ACL de red, las puertas de enlace de NAT y las conexiones entre pares.

Para configurar una conexión a Vertica:

1. En AWS Secrets Manager, cree un secreto con sus credenciales de Vertica, *verticaUsername* y *verticaPassword*. Para crear un secreto en Secrets Manager, siga el tutorial disponible en Crear un secreto AWS Secrets Manager en la documentación AWS Secrets Manager. Después de crear el secreto, conserve el nombre secreto, *secretName*, para el siguiente paso.

   - Al seleccionar pares clave/valor, genere un par para la clave `user` con el valor *verticaUsername*.

- Al seleccionar pares clave/valor, genere un par para la clave `password` con el valor *`verticaPassword`*.

2. En la consola de AWS Glue, genere una conexión mediante los pasos que se indican en the section called “Adición de una conexión de AWS Glue”. Tras crear la conexión, conserve el nombre de la conexión, *`connectionName`*, para el siguiente paso.

   - Al seleccionar un tipo de conexión, seleccione Vertica.
   - Al seleccionar Vertica Host, proporcione el nombre de host de la instalación de Vertica.
   - Al seleccionar Vertica Port, proporcione el portal a través del cual está disponible su instalación de Vertica.
   - Al seleccionar un secreto AWS, proporcione un *`secretName`*.

3. En las siguientes situaciones, es posible que necesite una configuración adicional:

   - Para las instancias de Vertica alojadas en AWS en una VPC de Amazon
     - Proporcione la información de conexión de Amazon VPC a la conexión de AWS Glue que define sus credenciales de seguridad de Vertica. Al crear o actualizar la conexión, configure los VPC, Subred y los grupos de seguridad en Opciones de red.

Deberá realizar los siguientes pasos antes de ejecutar su trabajo de AWS Glue:

- Otorga permisos al rol de IAM asociado al trabajo de AWS Glue para *`tempS3Path`*.
- Otorga permiso al rol de IAM asociado al trabajo de AWS Glue para leer el *`secretName`*.

Crear un nodo de origen de Vertica

Requisitos previos necesarios

- Una conexión de catálogo de datos de tipo AWS Glue Vertica, *`connectionName`* y una ubicación temporal de Amazon S3, *`tempS3Path`*, como se describe en la sección anterior, the section called “Crear una conexión Vertica”.
- *`Una tabla de Vertica de la que le gustaría leer, tableName o consultar targetQuery`*.

Agregar un origen de datos de Vertica

Para agregar un nodo de Origen de datos: Vertica:

1. Elija la conexión para el origen de datos de Vertica. Dado que lo ha creado, debería estar disponible en el menú desplegable. Si es necesario crear una conexión, elija Crear conexión de Vertica. Para obtener más información, consulte la sección the section called “Crear una conexión Vertica” anterior.

   Una vez que haya elegido una conexión, puede ver las propiedades de la conexión mediante un clic en Ver propiedades.
2. Elija la base de datos que contiene la tabla.
3. Elija el área de ensayo en Amazon S3 e introduzca un URI de S3A en *`tempS3Path`*.
4. Elija el origen de Vertica.

   - Elija una sola tabla: acceda a todos los datos de una sola tabla.
   - Ingresar una consulta personalizada: permite acceder a un conjunto de datos de varias tablas en función del consulta personalizada.
5. Si ha elegido una sola tabla, ingrese *`tableName`* y, de manera opcional, seleccione un esquema.

   Si eligió Introducir una consulta personalizada, introduzca una consulta SQL SELECT y, si lo desea, seleccione un esquema.
6. En las propiedades personalizadas de Vertica, ingrese los parámetros y valores según sea necesario.

Crear un nodo de destino de Vertica

Requisitos previos necesarios

- Una conexión de catálogo de datos de tipo AWS Glue Vertica, *`connectionName`* y una ubicación temporal de Amazon S3, *`tempS3Path`*, como se describe en la sección anterior, the section called “Crear una conexión Vertica”.

Agregar un destino de datos de Vertica

Para añadir un nodo de Destino de datos - Vertica:

1. Elija la conexión para el origen de datos de Vertica. Dado que lo ha creado, debería estar disponible en el menú desplegable. Si es necesario crear una conexión, elija Crear conexión de Vertica. Para obtener más información, consulte la sección the section called “Crear una conexión Vertica” anterior.

   Una vez que haya elegido una conexión, puede ver las propiedades de la conexión mediante un clic en Ver propiedades.
2. Elija la base de datos que contiene la tabla.
3. Elija el área de ensayo en Amazon S3 e introduzca un URI de S3A en *tempS3Path*.
4. Introduzca ***tableName*** y, si lo desea, seleccione un esquema.
5. En las propiedades personalizadas de Vertica, ingrese los parámetros y valores según sea necesario.

Opciones avanzadas

Puede brindar opciones avanzadas al crear un nodo de Vertica. Estas opciones son las mismas que las disponibles cuando se programa AWS Glue para scripts de Spark.

Consulte the section called “Conexiones Vertica”.

Uso de conectores y conexiones con AWS Glue Studio

AWS Glue proporciona soporte integrado con los almacenes de datos más utilizados (como Amazon Redshift, Amazon Aurora, Microsoft SQL Server, MySQL, MongoDB y PostgreSQL) mediante conexiones JDBC. AWS Glue también permite usar controladores JDBC personalizados en sus trabajos de extracción, transformación y carga (ETL). Para los almacenes de datos no soportados de forma nativa, como las aplicaciones de software como servicio (SaaS), puede utilizar conectores.

Un conector es un paquete de códigos opcionales que ayuda a acceder a almacenes de datos en AWS Glue Studio. Puede suscribirse a varios conectores ofrecidos en AWS Marketplace.

Al crear trabajos de ETL, puede utilizar un banco de datos compatible de forma nativa, un conector desde AWS Marketplace o sus propios conectores personalizados. Si utiliza un conector, primero debe crear una conexión para el conector. Una conexión contiene las propiedades necesarias para

conectarse a un almacén de datos determinado. Utilice la conexión con los orígenes de datos y los destinos de datos en el trabajo de ETL. Los conectores y las conexiones funcionan en conjunto para facilitar el acceso a los almacenes de datos.

Temas

- Información general del uso de conectores y conexiones
- Agregar conectores a AWS Glue Studio
- Conexiones disponibles
- Creación de conexiones para conectores
- Creación de trabajos con conectores personalizados
- Administración de conectores y conexiones
- Desarrollo de conectores personalizados
- Restricciones para el uso de conectores y conexiones en AWS Glue Studio

Información general del uso de conectores y conexiones

Una conexión contiene las propiedades necesarias para conectarse a un almacén de datos determinado. Cuando se crea una conexión, se almacena en el AWS Glue Data Catalog. Elija un conector y, a continuación, cree una conexión basada en ese conector.

Puede suscribirse a conectores para almacenes de datos no compatibles de forma nativa y AWS Marketplace, a continuación, utilizarlos al crear conexiones. Los desarrolladores también pueden crear sus propios conectores, que el usuario puede utilizar para crear conexiones.

> **Note**
>
> Las conexiones creadas mediante conectores personalizados o AWS Marketplace conectores AWS Glue Studio aparecen en la AWS Glue consola con el tipo establecido en. `UNKNOWN`

Los siguientes pasos describen el proceso general para utilizar conectores en AWS Glue Studio:

1. Suscríbase a un conector o desarrolle su propio conector y cárguelo enAWS Glue Studio. AWS Marketplace Para obtener más información, consulte Agregar conectores a AWS Glue Studio.

2. Revise la información de uso del conector. Puede encontrar esta información en la pestaña Usage (Uso) en la página de producto del conector. Por ejemplo, si haces clic en la pestaña Uso de la página de este producto, AWS GlueConnector para Google BigQuery, verás en la sección Recursos adicionales un enlace a un blog sobre el uso de este conector. Otros conectores pueden contener vínculos a las instrucciones de la sección Overview (Información general), como se muestra en la página de producto del conector para Conector Cloudwatch Logs para AWS Glue.
3. Cree una conexión. Puede elegir qué conector usar y proporcionar información adicional para la conexión, como credenciales de inicio de sesión, cadenas de URI e información de nube privada virtual (VPC). Para obtener más información, consulte Creación de conexiones para conectores.
4. Crear un rol de IAM para su trabajo. El flujo de trabajo asume los permisos de la IAM role (función de IAM) que especifica al crearla. Este rol de IAM debe tener permisos para autenticarse con sus almacenes de datos, extraer datos de sus almacén de datos y escribir datos a los mismos.
5. Cree un trabajo de ETL y configure las propiedades de origen de datos para su trabajo de ETL. Proporcione las opciones de conexión y la información de autenticación según las instrucciones del proveedor del conector personalizado. Para obtener más información, consulte Creación de trabajos con conectores personalizados.
6. Personalice su trabajo de ETL mediante al agregado de transformaciones o almacenes de datos adicionales, como se describe en Operaciones de ETL visuales con AWS Glue Studio.
7. Si utiliza un conector para el destino de datos, configure las propiedades de destino de datos para su trabajo de ETL. Proporcione las opciones de conexión y la información de autenticación según las instrucciones del proveedor del conector personalizado. Para obtener más información, consulte the section called “Creación de trabajos con conectores personalizados”.
8. Personalice el entorno de ejecución de trabajos mediante la configuración de las propiedades del trabajo, como se describe en Modificar las propiedades del trabajo.
9. Ejecute el trabajo.

Agregar conectores a AWS Glue Studio

Un conector es un fragmento de código que facilita la comunicación entre el almacén de datos y AWS Glue. Puedes suscribirte a un conector que se ofrece en AWS Marketplace o puedes crear tu propio conector personalizado.

Temas

- Suscribirse a conectores AWS Marketplace
- Creación de conectores personalizados

Suscribirse a conectores AWS Marketplace

AWS Glue Studiofacilita la adición de conectores desde AWS Marketplace.

Para añadir un conector de AWS Marketplace a AWS Glue Studio

1. En la consola de AWS Glue Studio, elija Conectores en el panel de navegación de la consola.
2. En la página Connectors (Conectores), elija Go to AWS Marketplace(Ir a MKT).
3. En AWS Marketplace, en Productos destacados, elija el conector que desee utilizar. Puede elegir uno de los conectores destacados o utilizar la búsqueda. Puede buscar el nombre o el tipo de conector, y puede utilizar opciones para refinar los resultados de búsqueda.

   Si desea utilizar uno de los conectores destacados, elija View product (Ver producto). Si utilizó la búsqueda para localizar un conector, elija el nombre del conector.
4. En la página de producto del conector, utilice las pestañas para ver información sobre el conector. Si decide comprar este conector, elija Continue to Subscribe (Continuar con la suscripción).
5. Proporcione la información de pago y, a continuación, elija Continue to Configure (Continuar con la configuración).
6. En la página Configure this software (Configurar este software), elija el método de implementación y la versión del conector que se va a utilizar. A continuación, elija Continue to Launch (Continuar con el lanzamiento).
7. En la página Launch this software (Lanzar este software), puede revisar las Usage Instructions (Instrucciones de uso) proporcionada por el proveedor del conector. Cuando esté listo para continuar, elija Activate connection in AWS Glue Studio (Activar conexión en AWS Glue Studio).

   Después de cierto tiempo, la consola muestra la página Create marketplace connection (Crear conexión de marketplace) en AWS Glue Studio.
8. Cree una conexión que utilice este conector, tal y como se describe en Creación de conexiones para conectores.

   También puede elegir Activate connector only (Activar sólo el conector) para omitir la creación de una conexión en este momento. Debe crear una conexión en una fecha posterior antes de poder usar el conector.

Creación de conectores personalizados

También puede crear su propio conector y después cargar el código en AWS Glue Studio.

Los conectores personalizados están integrados en AWS Glue Studio a través de la API de tiempo de ejecución de AWS Glue Spark. El tiempo de ejecución de AWS Glue Spark le permite conectar cualquier conector que cumpla con la interfaz de Spark, Athena o JDBC. Le permite transferir cualquier opción de conexión que esté disponible con el conector personalizado.

Puede encapsular todas sus propiedades de conexión con Conexiones de AWS Glue y proporcionar el nombre de la conexión para su trabajo de ETL. La integración con las conexiones del Catálogo de datos le permite utilizar las mismas propiedades de conexión en varias llamadas en una sola aplicación Spark o entre diferentes aplicaciones.

Puede especificar opciones adicionales para la conexión. El script de trabajo que genera AWS Glue Studio contiene una entrada de `Datasource` que utiliza la conexión para conectar el conector con las opciones especificadas. Por ejemplo:

```
Datasource = glueContext.create_dynamic_frame.from_options(connection_type =
"custom.jdbc", connection_options = {"dbTable":"Account","connectionName":"my-custom-
jdbc-
connection"}, transformation_ctx = "DataSource0")
```

Para agregar un conector personalizado a AWS Glue Studio

1. Cree el código para su conector personalizado. Para obtener más información, consulte Desarrollo de conectores personalizados.
2. Agregue soporte para características de AWS Glue a su conector. Aquí hay algunos ejemplos de estas características y cómo se utilizan dentro del script de trabajo generado por AWS Glue Studio:

   - Mapeo de tipos de datos: el conector puede convertir el tipo de las columnas mientras las lee desde el almacén de datos subyacente. Por ejemplo, un `dataTypeMapping` de `{"INTEGER":"STRING"}` convierte todas las columnas de tipo `Integer` a columnas de tipo `String` al analizar los registros y construir el archivo `DynamicFrame`. Esto ayuda a los usuarios a convertir columnas en los tipos de su elección.

     ```
     DataSource0 = glueContext.create_dynamic_frame.from_options(connection_type
     = "custom.jdbc", connection_options = {"dataTypeMapping":{"INTEGER":"STRING"}",
     ```

```
connectionName":"test-connection-jdbc"}, transformation_ctx = "DataSource0")
```

- Partición para lecturas paralelas: AWS Glue permite leer datos paralelos desde el almacén de datos particionando los datos en una columna. Debe especificar la columna de partición, el límite inferior de partición, el límite superior de partición y el número de particiones. Esta función le permite hacer uso del paralelismo de datos y de varios ejecutores Spark asignados para la aplicación Spark.

```
DataSource0 = glueContext.create_dynamic_frame.from_options(connection_type
= "custom.jdbc", connection_options = {"upperBound":"200","numPartitions":"4",
"partitionColumn":"id","lowerBound":"0","connectionName":"test-connection-jdbc"},
transformation_ctx = "DataSource0")
```

- Úselo AWS Secrets Manager para almacenar credenciales: la conexión al catálogo de datos también puede contener un `secretId` formulario secreto almacenado en AWS Secrets Manager. El AWS secreto puede almacenar de forma segura la información de autenticación y credenciales y proporcionársela AWS Glue en tiempo de ejecución. También puede especificar el `secretId` del script de Spark de la siguiente manera:

```
DataSource = glueContext.create_dynamic_frame.from_options(connection_type
= "custom.jdbc", connection_options = {"connectionName":"test-connection-jdbc",
 "secretId"-> "my-secret-id"}, transformation_ctx = "DataSource0")
```

- Filtrar los datos de origen con predicados de fila y proyecciones de columna: el tiempo de ejecución de AWS Glue Spark también permite a los usuarios insertar consultas SQL para filtrar datos en el origen con predicados de fila y proyecciones de columna. Esto permite que su trabajo de ETL cargue datos filtrados más rápidamente desde los almacenes de datos que admiten inserciones. Un ejemplo de consulta SQL insertada a un origen de datos JDBC es: `SELECT id, name, department FROM department WHERE id < 200.`

```
DataSource = glueContext.create_dynamic_frame.from_options(connection_type =
"custom.jdbc", connection_options = {"query":"SELECT id, name, department FROM
 department
WHERE id < 200","connectionName":"test-connection-jdbc"}, transformation_ctx =
"DataSource0")
```

- Marcadores de trabajo: AWS Glue admite la carga progresiva de datos de orígenes JDBC. AWS Glue realiza un seguimiento del último registro procesado del almacén de datos y procesa nuevos registros de datos en las siguientes ejecuciones de trabajos de ETL. Los marcadores de trabajo utilizan la clave principal como columna predeterminada para la clave

de marcador, siempre que esta columna aumente o disminuya en forma secuencial. Para obtener más información acerca de los marcadores de trabajo, consulte Marcadores de trabajo en la Guía para desarrolladores de AWS Glue .

```
DataSource0 = glueContext.create_dynamic_frame.from_options(connection_type =
"custom.jdbc", connection_options = {"jobBookmarkKeys":["empno"],
 "jobBookmarkKeysSortOrder"
:"asc", "connectionName":"test-connection-jdbc"}, transformation_ctx =
 "DataSource0")
```

3. Empaquete el conector personalizado como un archivo JAR y cárguelo a Amazon S3.
4. Pruebe su conector personalizado. Para obtener más información, consulte las instrucciones GitHub en la guía Glue Custom Connectors: Local Validation Tests Guide.
5. En la consola de AWS Glue Studio, elija Connectors (Conectores) en el panel de navegación de la consola.
6. En la página Connectors (Conectores), seleccione Create custom connector (Crear conector personalizado).
7. En la página Create custom connector (Crear conector personalizado), ingrese la siguiente información:
   - La ruta a la ubicación del archivo JAR de código personalizado en Amazon S3.
   - Un nombre para el conector que utilizará AWS Glue Studio.
   - El tipo de conector, que puede ser JDBC, Spark o Athena.
   - El nombre del punto de entrada dentro del código personalizado que AWS Glue Studio invoca para utilizar el conector.
     - Para los conectores JDBC, este campo debe ser el nombre de clase del controlador JDBC.
     - En el caso de los conectores Spark, este campo debe ser el nombre de clase del origen de datos completo, o su alias, que se utiliza al cargar el origen de datos Spark con el operador de `format`.
   - (Sólo JDBC) la dirección URL base utilizada por la conexión JDBC para el almacén de datos.
   - (Opcional) descripción del conector personalizado.
8. Elija Create connector (Crear conector).
9. Desde la página Connectors (Conectores), cree una conexión que utilice este conector, tal y como se describe en Creación de conexiones para conectores.

Conexiones disponibles

Las siguientes conexiones están disponibles al crear conexiones para conectores:

- Amazon Aurora: un motor de base de datos relacional escalable y de alto rendimiento con seguridad, copia de seguridad y restauración integradas y aceleración en memoria.
- Amazon DocumentDB: un servicio de base de datos de documentos escalable, de alta disponibilidad y totalmente gestionado que admite las API de MongoDB y SQL.
- Amazon Redshift: un servicio de base de datos de documentos escalable, de alta disponibilidad y totalmente administrado que admite las API de MongoDB y SQL.
- Azure SQL: un servicio de base de datos relacional basado en la nube de Microsoft Azure que proporciona capacidades de administración y almacenamiento de datos escalables, fiables y seguras.
- Cosmos DB: un servicio de base de datos en la nube distribuido a nivel mundial de Microsoft Azure que proporciona capacidades de consulta y almacenamiento de datos escalables y de alto rendimiento.
- Google BigQuery: un almacén de datos en la nube sin servidor para ejecutar consultas SQL rápidas en conjuntos de datos de gran tamaño.
- JDBC: un sistema de administración de base de datos relacional (RDBMS) que utiliza una API de Java para conectarse e interactuar con las conexiones de datos.
- Kafka: una plataforma de procesamiento de flujos de código abierto que se utiliza para el flujo de datos y la mensajería en tiempo real.
- MariaDB: una bifurcación de MySQL desarrollada por la comunidad que ofrece rendimiento, escalabilidad y características mejorados.
- MongoDB: una base de datos multiplataforma orientada a documentos que proporciona alta escalabilidad, flexibilidad y rendimiento.
- MongoDB Atlas: una oferta de base de datos como servicio (DBaaS) basada en la nube de MongoDB que simplifica la administración y el escalado de las implementaciones de MongoDB.
- Microsoft SQL Server: un sistema de administración de base de datos relacional (RDBMS) de Microsoft que proporciona sólidas capacidades de almacenamiento, análisis e informes de datos.
- MySQL: un sistema de administración de bases de datos relacionales (RDBMS) de código abierto que se usa ampliamente en aplicaciones web y es conocido por su fiabilidad y escalabilidad.
- Red: un origen de datos de red representa un recurso o servicio accesible desde la red al que se puede acceder mediante una plataforma de integración de datos.

- OpenSearch— una fuente OpenSearch de datos es una aplicación desde la que OpenSearch se pueden conectar e ingerir datos.
- Oracle: un sistema de administración de base de datos relacional (RDBMS) de Oracle Corporation que proporciona sólidas capacidades de almacenamiento, análisis e informes de datos.
- PostgreSQL: un sistema de administración de base de datos relacional (RDBMS) de código abierto que proporciona sólidas capacidades de almacenamiento, análisis e informes de datos.
- Salesforce: Salesforce proporciona un software de gestión de relaciones con los clientes (CRM) que lo ayuda con las ventas, el servicio al cliente, el comercio electrónico y más. Si es usuario de Salesforce, puede conectarse AWS Glue a su cuenta de Salesforce. A continuación, puede utilizar Salesforce como fuente o destino de datos en sus trabajos de ETL. Ejecute estos trabajos para transferir datos entre Salesforce y AWS los servicios u otras aplicaciones compatibles.
- SAP HANA: una plataforma de análisis y base de datos en memoria que proporciona un procesamiento rápido de datos, análisis avanzados e integración de datos en tiempo real.
- Snowflake: un almacén de datos basado en la nube que proporciona servicios de análisis y almacenamiento de datos escalables y de alto rendimiento.
- Teradata: un sistema de administración de base de datos relacional (RDBMS) que proporciona capacidades de almacenamiento, análisis e informes de datos de alto rendimiento.
- Vertica: almacenamiento de datos analíticos orientado a columnas diseñado para el análisis de macrodatos que ofrece un rendimiento de consultas rápido, análisis avanzados y escalabilidad.

Creación de conexiones para conectores

Una conexión a AWS Glue es un objeto de Data Catalog que almacena información de conexión para un almacén de datos determinado. Las conexiones almacenan credenciales de inicio de sesión, cadenas de URI, información de nube privada virtual (VPC), etc. Al crear conexiones en el Data Catalog se ahorra el esfuerzo de tener que especificar todos los detalles de conexión cada vez que se crea un trabajo.

Para crear una conexión para un conector

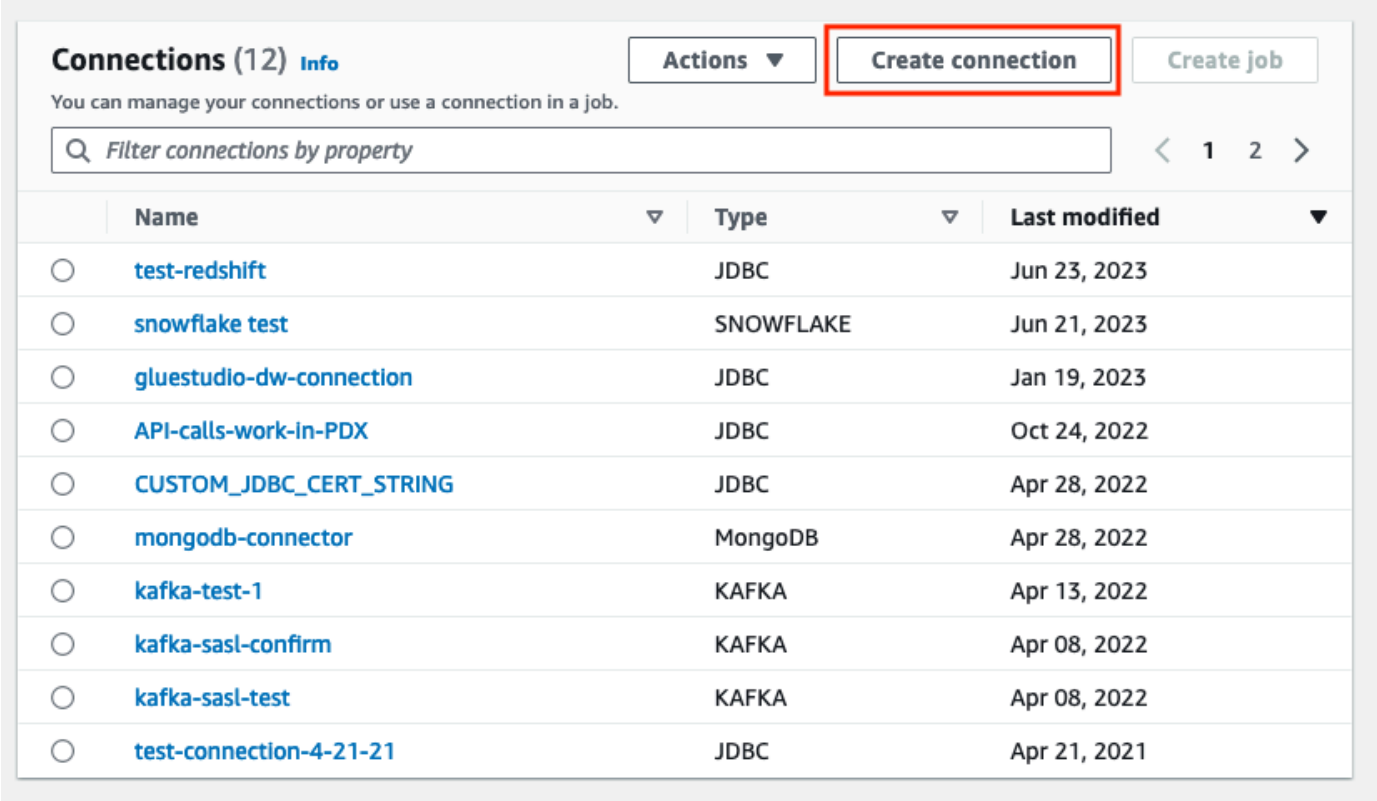
1. En la consola de AWS Glue Studio, elija Conectores en el panel de navegación de la consola. En la sección Conexiones, elija Crear conexión.
2. Elija el origen de datos para la que desee crear una conexión en el paso 1 del asistente de Creación de conexiones de datos. Existen varias maneras de ver los orígenes de datos disponibles, como:

- Filtrar los orígenes de datos disponibles seleccionando una pestaña. De forma predeterminada, está seleccionada la opción Todos los conectores.
- Seleccione Lista para ver los orígenes de datos en forma de lista o vuelva a Cuadrícula para ver los conectores disponibles en el diseño de cuadrícula.
- Utilice la barra de búsqueda para reducir la lista de orígenes de datos. A medida que escribe, se muestran las coincidencias de búsqueda y los orígenes que no coinciden se eliminan de la vista.

Una vez que haya elegido el origen de datos, elija Siguiente.

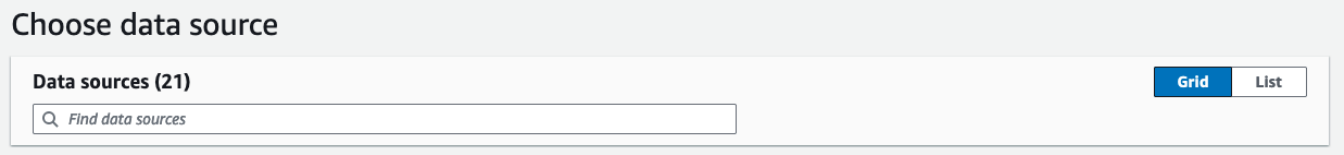
3. Configure la conexión en el paso 2 del asistente.

   Ingrese los detalles de conexión. En función del tipo de conector que haya seleccionado, se le pedirá que ingrese información adicional:



4. Elija el origen de datos para la que desee crear una conexión en el paso 1 del asistente de Creación de conexiones de datos. Existen varias maneras de ver los orígenes de datos disponibles. De forma predeterminada, verá todos los orígenes de datos disponibles en un diseño de cuadrícula. También puede:

- Seleccione Lista para ver los orígenes de datos en forma de lista o vuelva a Cuadrícula para ver los conectores disponibles en el diseño de cuadrícula.
- Utilice la barra de búsqueda para reducir la lista de orígenes de datos. A medida que escribe, se muestran las coincidencias de búsqueda y los orígenes que no coinciden se eliminan de la vista.

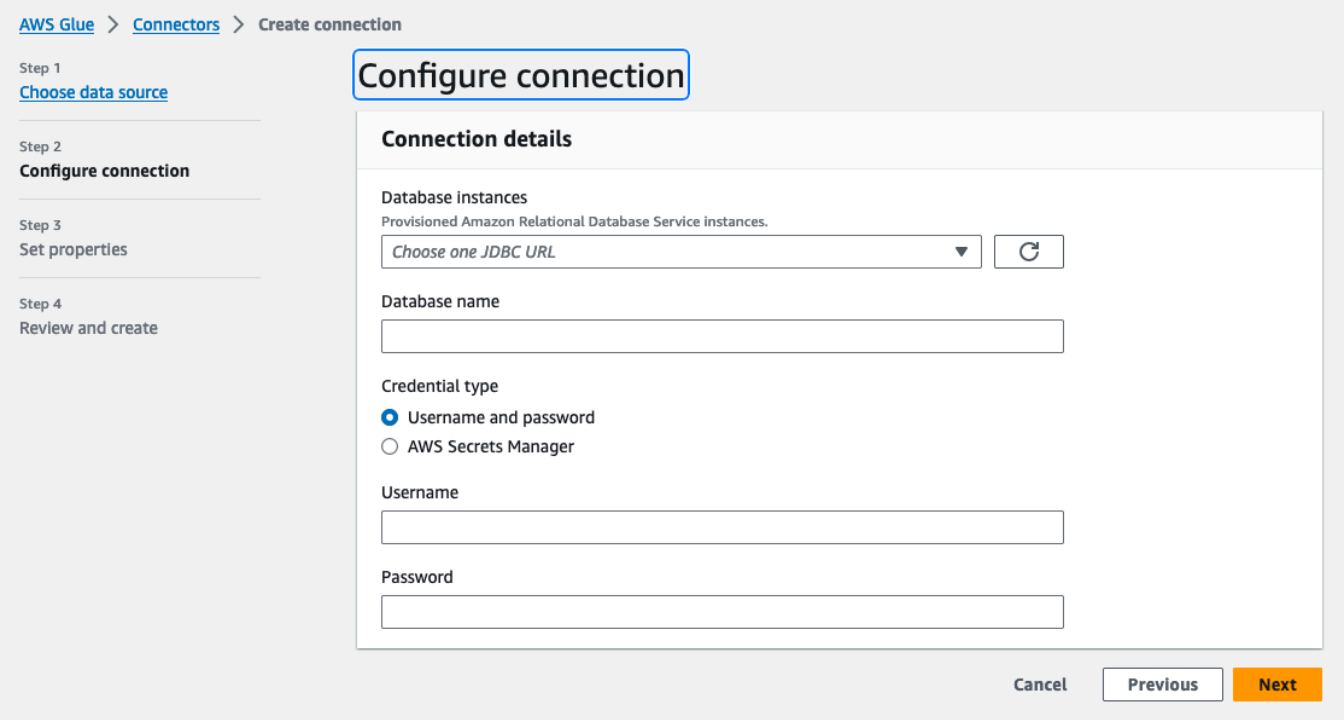


Una vez que haya elegido el origen de datos, elija Siguiente.

5. Configure la conexión en el paso 2 del asistente.

   Ingrese los detalles de conexión. En función del tipo de conector que haya seleccionado, se le pedirá que ingrese información adicional de la conexión. Esto puede incluir:

   - Detalles de la conexión: estos campos cambiarán en función del origen de datos a la que se conecte. Por ejemplo, si se conecta a las bases de datos de Amazon DocumentDB, introducirá la URL de Amazon DocumentDB. Si se está conectando a Amazon Aurora, elegirá la instancia de la base de datos e ingresará el nombre de la base de datos. Los siguientes son los detalles de conexión necesarios para Amazon Aurora:



- Tipo de credencial: elija entre Nombre de usuario y contraseña o AWS Secrets Manager. Introduzca la información de autenticación solicitada.
- Para los conectores que utilizan JDBC, ingrese la información necesaria para crear la dirección URL JDBC para el almacén de datos.
- Si usa una nube privada virtual (VPC), ingrese la información de red de su VPC.

6. Establezca las propiedades de conexión en el paso 3 del asistente. Puede añadir una descripción y etiquetas como parte opcional de este paso. El nombre es obligatorio y viene rellenado previamente con un valor predeterminado. Elija Siguiente.
7. Revise el origen, los detalles y las propiedades de la conexión. Seleccione Editar para realizar cambios en cualquiera de los pasos del asistente. Cuando esté listo, elija Crear conexión.

   Elija Crear conexión.

   Será dirigido a la página Conectores y el banner informativo le indicará la conexión que se creó. Ahora puede utilizar la conexión en los trabajos de AWS Glue Studio.

Creación de una conexión Kafka

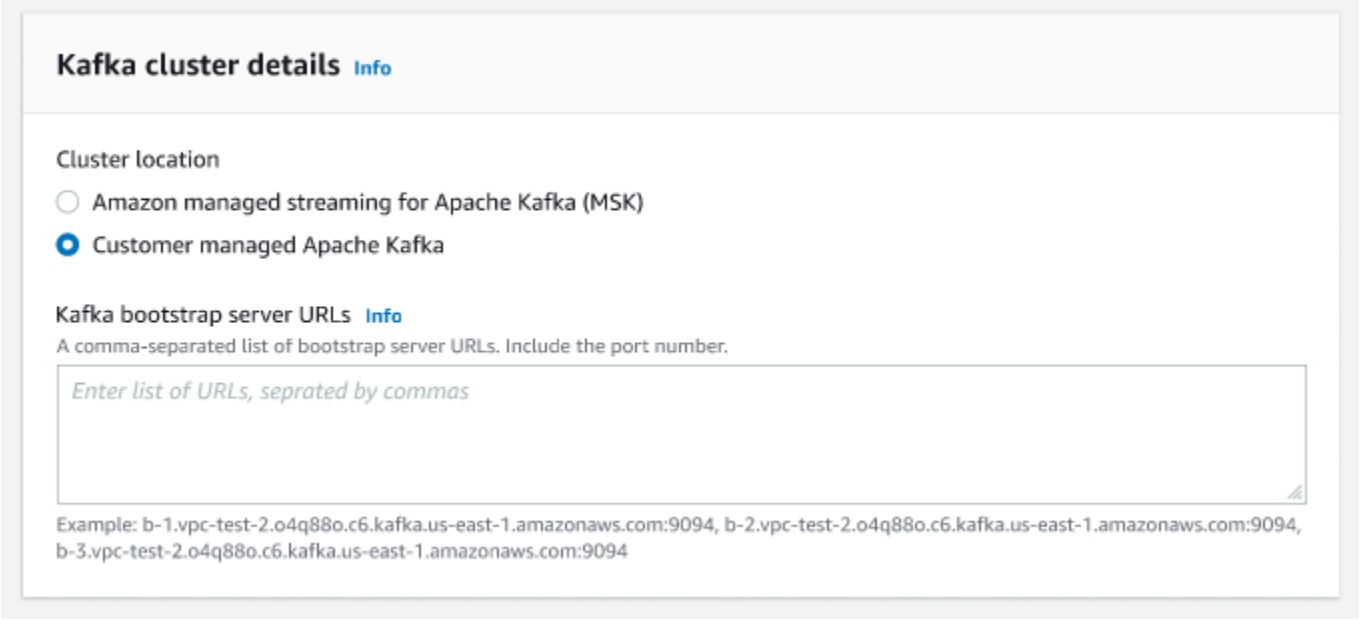
Al crear una conexión Kafka, si se selecciona la opción Kafka en el menú desplegable, se mostrarán configuraciones adicionales que puede establecer:

- Detalles del clúster de Kafka
- Autenticación
- Cifrado
- Opciones de red

Configuración del clúster de Kafka

1. Elija la ubicación del clúster. Puede elegir entre un streaming administrado de Amazon para un clúster Apache Kafka (MSK o un clúster Apache Kafka administrado por el cliente. Para obtener más información sobre streaming administrado de Amazon para Apache Kafka, consulte Amazon managed streaming for Apache Kafka (MSK).

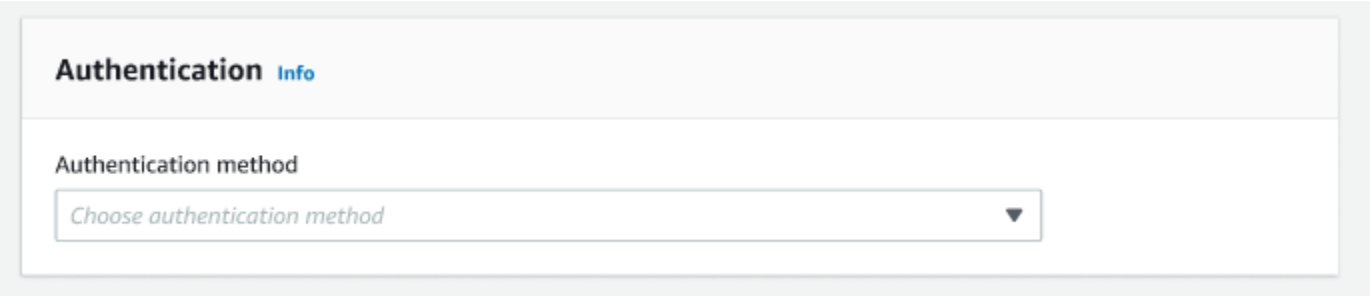
   > **Note**
   >
   > Amazon Managed Streaming for Apache Kafka solo admite métodos de autenticación TLS y SASL/SCRAM-SHA-512.



2. Ingrese las URL de sus servidores Bootstrap de Kafka. Puede ingresar más de uno separando cada servidor por una coma. Incluya el número de puerto al final de la URL agregando `:<port number>`.

   Por ejemplo: `b-1.vpc-test-2.034a88o.kafka-us-east-1.amazonaws.com:9094`.

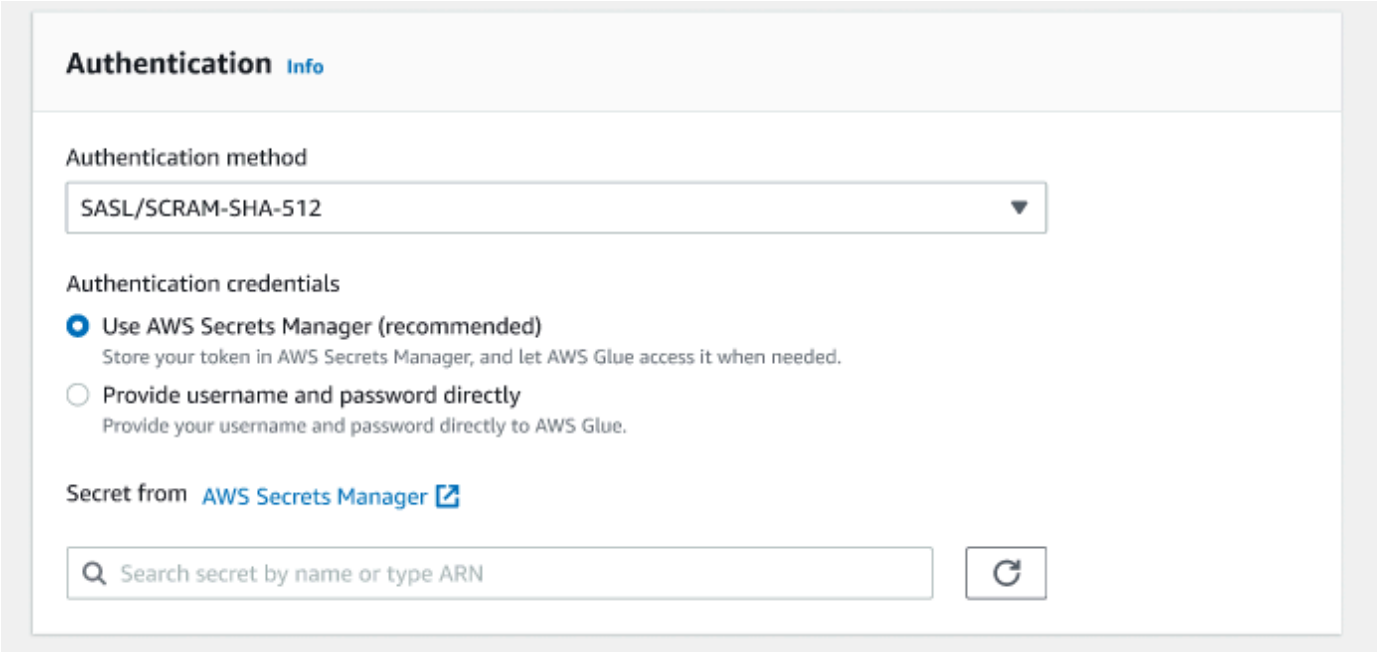
Seleccione un método de autenticación



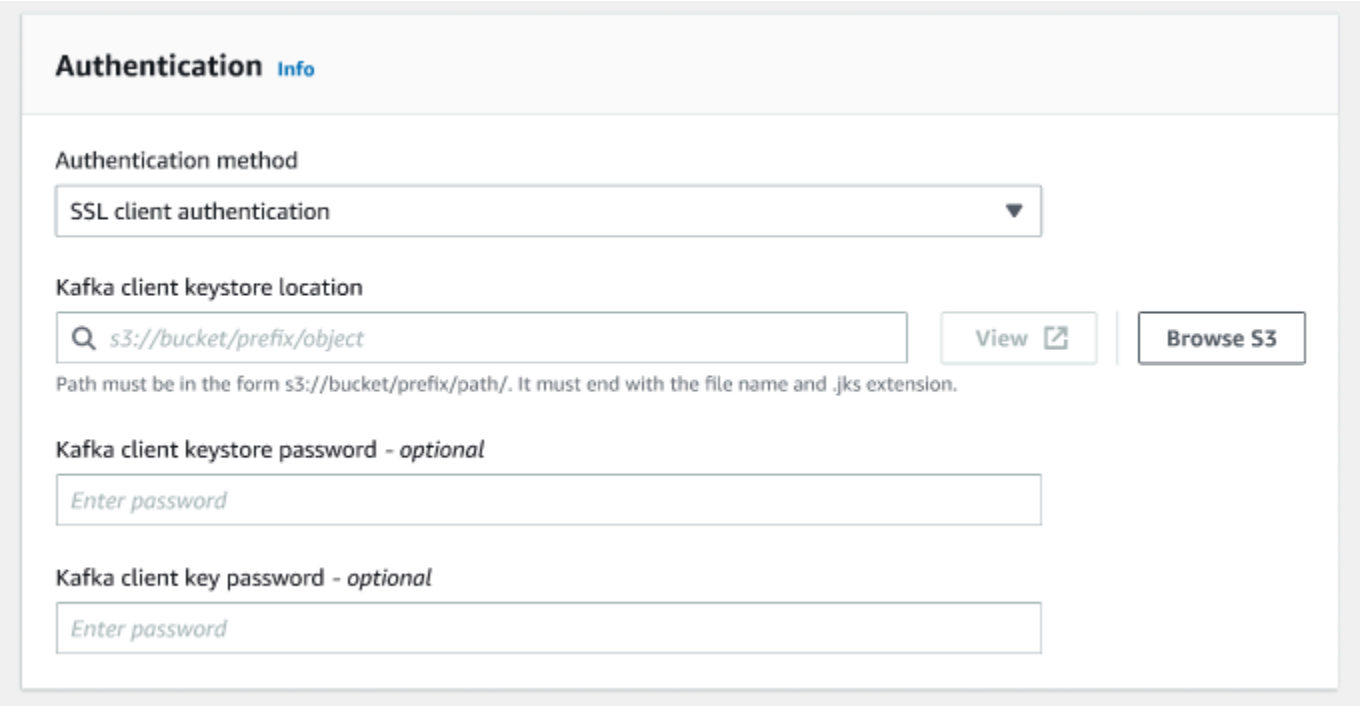
AWS Glue admite el marco de autenticación simple y capa de seguridad (SASL) para la autenticación. El esquema SASL es compatible con distintos mecanismos de autenticación; AWS Glue ofrece los protocolos SCRAM (nombre de usuario y contraseña), GSSAPI (protocolo Kerberos) y PLAIN (nombre de usuario y contraseña).

Al elegir un método de autenticación en el menú desplegable, se pueden seleccionar los siguientes métodos de autenticación de clientes:

- Ninguno: sin autenticación. Esto resulta útil si crea una conexión con fines de prueba.
- SASL/SCRAM-SHA-512: elija este método de autenticación para especificar credenciales de autenticación. Existen dos opciones disponibles:
  - Utilizar AWS Secrets Manager (recomendado): si selecciona esta opción, puede almacenar sus credenciales en AWS Secrets Manager y dejar que AWS Glue acceda a la información cuando sea necesario. Especificar el secreto donde están almacenadas las credenciales de autenticación SSL o SASL.



    - Brinde de manera directa un usuario y una contraseña.
- SASL/GSSAPI (Kerberos): si selecciona esta opción, puede seleccionar la ubicación del archivo keytab, el archivo krb5.conf e ingresar el nombre principal y el nombre del servicio de Kerberos. Las ubicaciones de los archivos keytab y krb5.conf deben estar en una ubicación de Simple Storage Service (Amazon S3). Dado que MSK aún no admite SASL/GSSAPI, esta opción solo está disponible para clústeres Apache Kafka administrados por el cliente. Para obtener más información, consulte MIT Kerberos Documentation: Keytab (Documentación de MIT Kerberos: Keytab).
- SASL/PLAIN: seleccione este método de autenticación para especificar las credenciales de autenticación. Existen dos opciones disponibles:
    - Utilizar AWS Secrets Manager (recomendado): si selecciona esta opción, puede almacenar sus credenciales en AWS Secrets Manager y dejar que AWS Glue acceda a la información cuando sea necesario. Especificar el secreto donde están almacenadas las credenciales de autenticación SSL o SASL.
    - Brinde de manera directa un usuario y una contraseña.
- Autenticación de cliente SSL: si selecciona esta opción, puede seleccionar la ubicación del almacén de claves del cliente Kafka navegando por Simple Storage Service (Amazon S3). Opcionalmente, puede ingresar la contraseña del almacén de claves del cliente Kafka y la contraseña de clave de cliente Kafka.

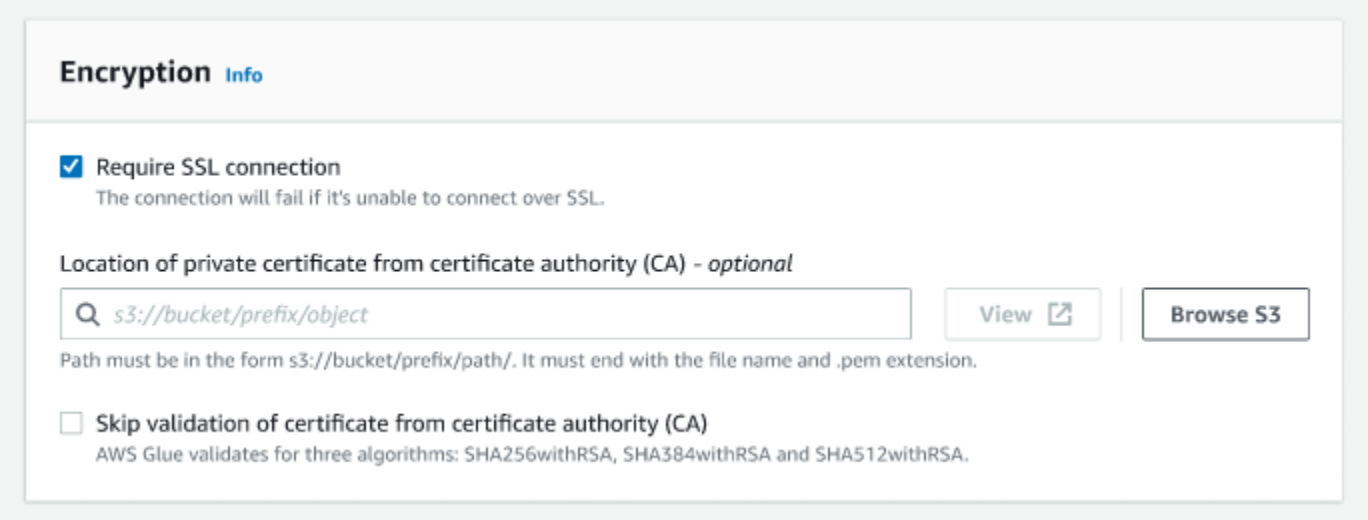


Establecimiento de la configuración de cifrado

1. Si la conexión Kafka requiere una conexión SSL, active la casilla de verificación de Require SSL connection (Requerir conexión SSL). Tenga en cuenta que la conexión fallará si no puede conectarse a través de SSL. El SSL para el cifrado es opcional y puede utilizarse con cualquier método de autenticación (como SASL/SCRAM-SHA-512, SASL/GSSAPI, SASL/PLAIN o SSL Client Authentication).

   Si el método de autenticación se establece en SSL client authentication (Autenticación de cliente SSL), esta opción se seleccionará automáticamente y se desactivará para evitar cambios.
2. (Opcional). Elija la ubicación del certificado privado de la entidad de certificación (CA). Tenga en cuenta que la ubicación de la certificación debe estar en una ubicación de S3. Elija la opción Browse (Explorar) para elegir el archivo de un bucket de S3 conectado. La ruta debe tener el formato `s3://bucket/prefix/filename.pem`. Debe terminar con el nombre de archivo y la extensión .pem.
3. Puede optar por omitir la validación del certificado de una entidad de certificación (CA). Elija la casilla de verificación Skip validation of certificate from certificate authority (CA) (Omitir la validación del certificado de la entidad de certificación [CA]). Si esta casilla no está marcada, AWS Glue valida certificados para tres algoritmos:

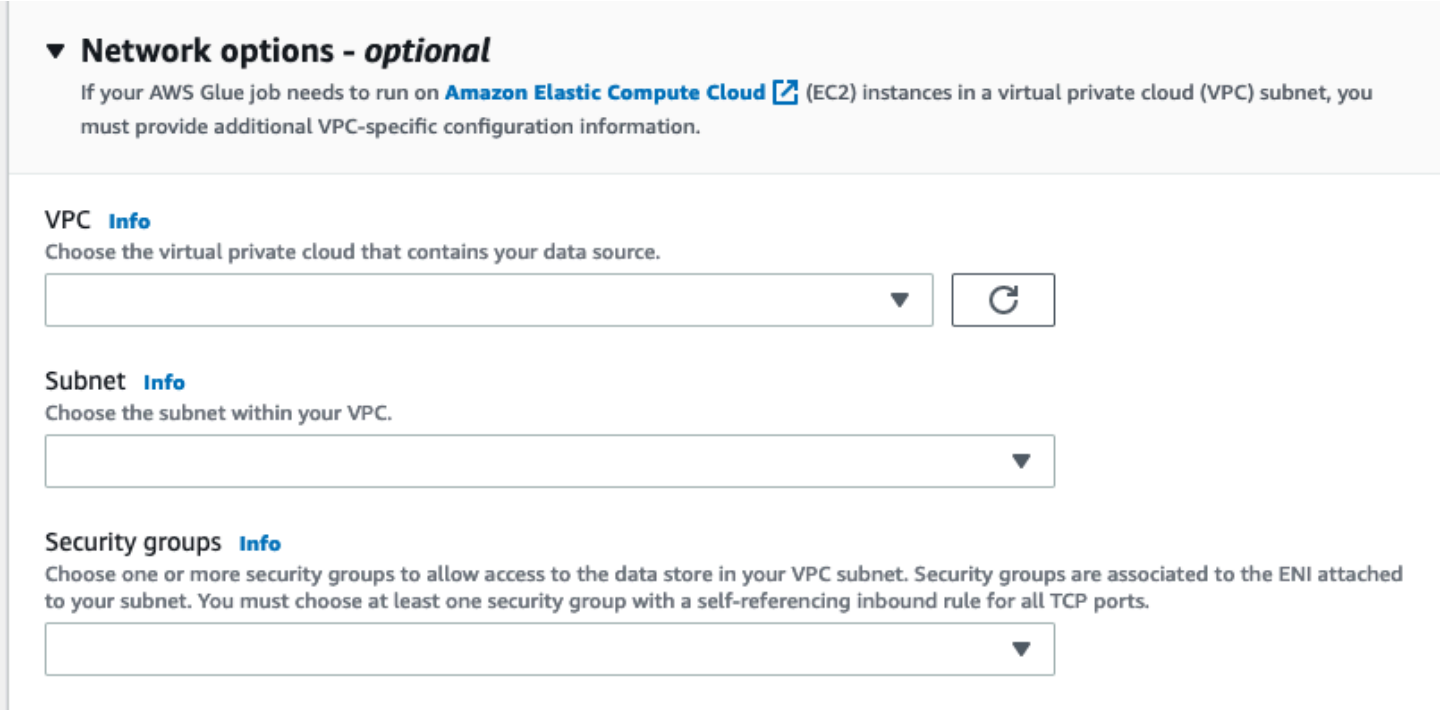
   - SHA256withRSA
   - SHA384withRSA
   - SHA512withRSA



(Opcional) Opciones de red

A continuación, se indican pasos opcionales para configurar la VPC, subred y grupos de seguridad. Si el trabajo de AWS Glue debe ejecutarse en instancias de Amazon EC2 en una subred de nube virtual privada (VPC), debe proporcionar información adicional de configuración específica de la VPC.

1. Elija el nombre de la nube virtual privada (VPC) que contenga el origen de datos.
2. Elija una subred con su VPC.
3. Elija uno o más grupos de seguridad que permitan el acceso al almacén de datos de la subred VPC. Los grupos de seguridad están asociados a la ENI asociada a la subred. Debe elegir al menos un grupo de seguridad con una regla de entrada con autorreferencia para todos los puertos TCP.



Creación de trabajos con conectores personalizados

Puede utilizar conectores y conexiones tanto para nodos de origen de datos como para nodos de destino de datos en AWS Glue Studio.

Temas

- Creación de trabajos que utilicen un conector para el origen de datos
- Configuración de las propiedades de origen para los nodos que utilizan conectores

- Configuración de las propiedades de destino para los nodos que utilizan conectores

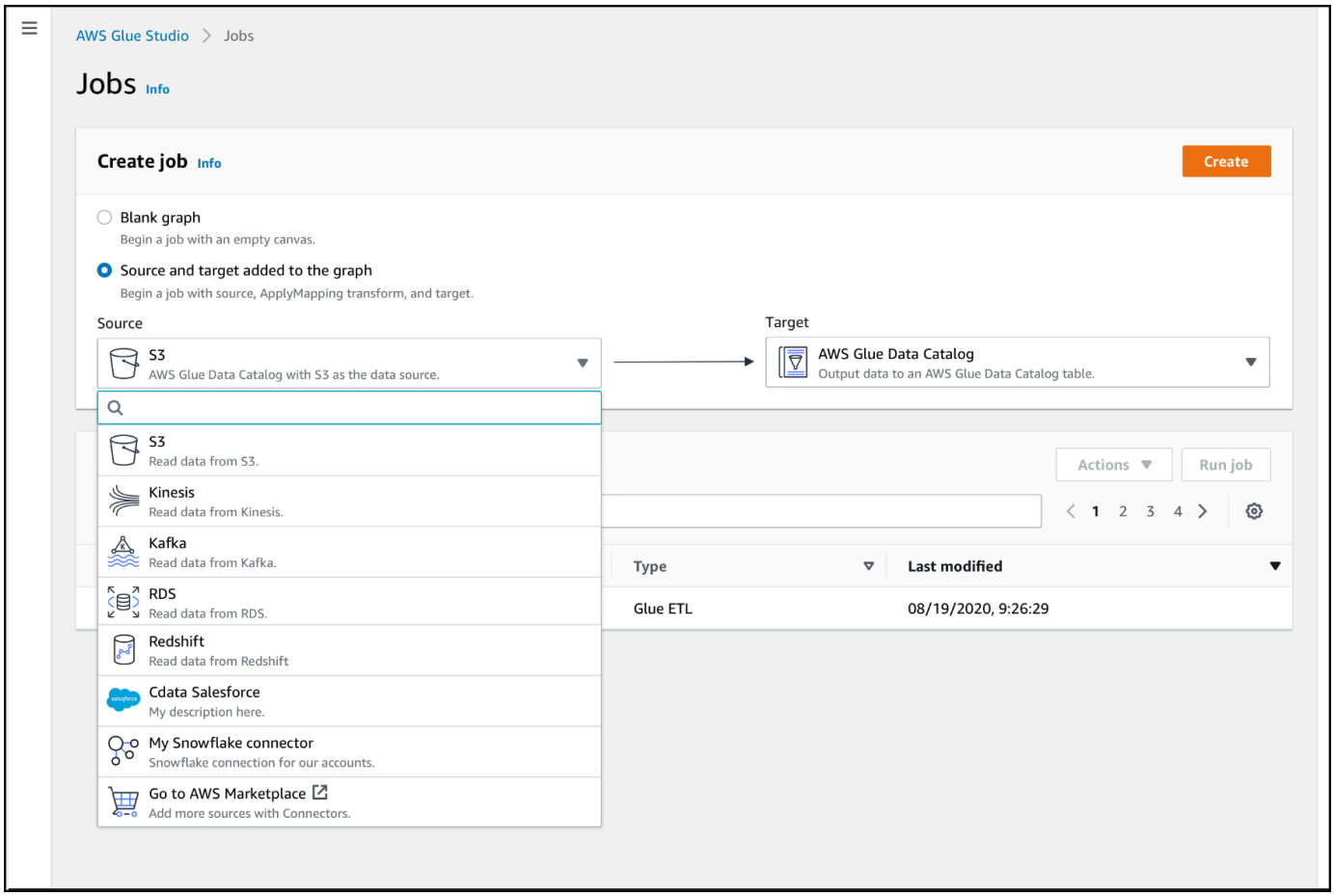
Creación de trabajos que utilicen un conector para el origen de datos

Al crear un nuevo trabajo, puede elegir un conector para el origen de datos y los destinos de datos.

Para crear un trabajo que utilice conectores para el origen de datos o el destino de datos

1. Inicie sesión en la AWS Glue Studio consola AWS Management Console y ábrala en https://console.aws.amazon.com/gluestudio/.
2. En la página Connectors (Conectores), en la lista de recursos Your connections (Sus conexiones), elija la conexión que desea usar en su trabajo y, a continuación, elija Create job (Crear el trabajo).

   Como alternativa, en la página Jobs (Trabajos) de AWS Glue Studio, en Create job (Crear el trabajo), elija Source and target added to the graph (Origen y destino agregados al gráfico). En la lista desplegable Source (Origen), elija el conector personalizado que desea usar en el trabajo. También puede elegir un conector para Target (Destino).



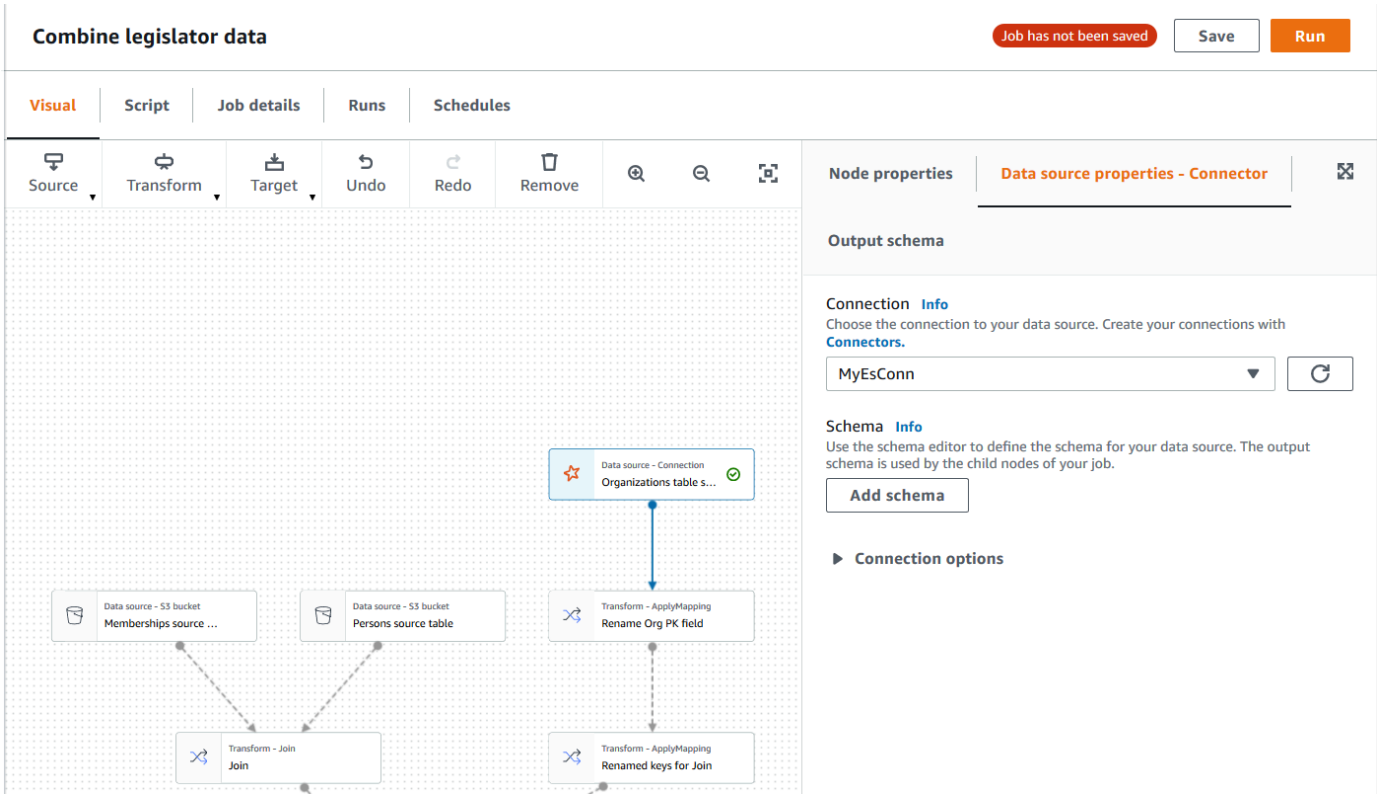
3. Elija Create (Crear) para abrir el editor visual de trabajos.
4. Configure el nodo de origen de datos, como se describe en Configuración de las propiedades de origen para los nodos que utilizan conectores.
5. Continúe creando su trabajo de ETL mediante el agregado de transformaciones, almacenes de datos adicionales y destinos de datos, como se describe en Operaciones de ETL visuales con AWS Glue Studio.
6. Personalice el entorno de ejecución del trabajo mediante la configuración de las propiedades del trabajo, como se describe en Modificar las propiedades del trabajo.
7. Guarde y ejecute el trabajo.

Configuración de las propiedades de origen para los nodos que utilizan conectores

Después de crear un trabajo que utiliza un conector para el origen de datos, el editor visual de trabajos muestra un gráfico de trabajo con un nodo de origen de datos configurado para el conector. Debe configurar las propiedades del origen de datos para ese nodo.

Para configurar las propiedades de un nodo de origen de datos que utiliza un conector

1. Elija el nodo de origen de datos del conector en el gráfico de trabajo o agregue un nodo nuevo y elija el conector para el Node type (Tipo de nodo). A continuación, en el lado derecho, en el panel de detalles del nodo, seleccione la pestaña Data source properties (Propiedades de origen de datos), si aún no está seleccionada.



2. En la pestaña Data source properties (Propiedades de origen de datos), elija la conexión que desea utilizar para este trabajo.

   Ingrese la información adicional necesaria para cada tipo de conexión:

   JDBC

   - Data source input type (Tipo de entrada de origen de datos): elija proporcionar un nombre de tabla o una consulta SQL como origen de datos. En función de su elección, deberá proporcionar la siguiente información adicional:
     - Table name (Nombre de la tabla): el nombre de la tabla en el origen de datos. Si la fuente de datos no utiliza la tabla de términos, proporcione el nombre de la estructura de datos adecuada, tal y como se indica en la información de uso del conector personalizado (que está disponible en AWS Marketplace).
     - Filter predicate (Filtrar predicado): una cláusula de condición que se usa al leer el origen de datos, similar a `WHERE`, utilizada para recuperar un subconjunto de los datos.
     - Query code (Código de consulta): ingrese una consulta SQL que se utilizará para recuperar un conjunto de datos específico del origen de datos. Un ejemplo de una consulta SQL básica es:

```
SELECT column_list FROM
                          table_name WHERE where_clause
```

- Schema (Esquema): ya que AWS Glue Studio utiliza la información almacenada en la conexión para tener acceso al origen de datos en lugar de recuperar información de los metadatos de una tabla del Catálogo de datos, debe proporcionar los metadatos del esquema para el origen de datos. Elija Add schema (Agregar esquema) para abrir el editor de esquemas.

  Para obtener instrucciones sobre cómo utilizar el editor de esquemas, consulte Edición de esquema para un nodo de transformación personalizado.

- Partition column (Columna de partición): (opcional) puede optar por particionar las lecturas de datos al proporcionar valores para Partition column (Columna de partición), Lower bound (Límite inferior),Upper bound (Límite superior) y Number of partitions (Número de particiones).

  Los valores `lowerBound` y `upperBound` se utilizan para decidir el intervalo de partición, no para filtrar las filas de la tabla. Todas las filas de la tabla se particionan y se devuelven.

  > **Note**
  >
  > La partición de columnas agrega una condición de partición adicional a la consulta utilizada para leer los datos. Cuando se utiliza una consulta en lugar de un nombre de tabla, debe validar que la consulta funciona con la condición de partición especificada. Por ejemplo:
  >
  > - Si el formato de consulta es `"SELECT col1 FROM table1"`, pruebe la consulta al agregar una cláusula `WHERE` al final de la consulta que utiliza la columna de partición.
  > - Si su formato de consulta es `"SELECT col1 FROM table1 WHERE col2=val"`, pruebe la consulta al ampliar la cláusula `WHERE` con `AND` y una expresión que utiliza la columna de partición.

- Data type casting (Conversión de tipo de datos): si el origen de datos utiliza tipos de datos que no están disponibles en JDBC, utilice esta sección para especificar cómo se debe convertir un tipo de datos del origen de datos en tipos de datos JDBC. Puede especificar

hasta 50 conversiones de tipos de datos diferentes. Todas las columnas del origen de datos que utilizan el mismo tipo de datos se convierten de la misma manera.

Por ejemplo, si tiene tres columnas en el origen de datos que utilizan el tipo de datos `Float` e indica que el tipo de datos `Float` se debe convertir al tipo de datos `String` de JDBC, las tres columnas que utilizan el tipo de datos `Float` se convierten a los tipos de datos `String`.

- Job bookmark keys (Claves de marcadores de trabajo): los marcadores de trabajo ayudan a AWS Glue a mantener la información de estado y evitar el reprocesamiento de los datos antiguos. Especifique una o más columnas como claves favoritas. AWS Glue Studio utiliza claves favoritas para realizar un seguimiento de los datos que ya se han procesado durante una ejecución anterior del trabajo de ETL. Cualquier columna que utilice para claves de marcadores personalizadas debe ser estrictamente monotónica en aumento o disminución, pero se permiten espacios.

  Si ingresa varias claves de marcadores, se combinan para formar una única clave compuesta. Una clave de marcador de trabajo compuesta no debe contener columnas duplicadas. Si no especifica ninguna clave favorita, AWS Glue Studio utiliza la clave principal como clave favorita de forma predeterminada, siempre que aumente o disminuya en forma secuencial (sin brechas). Si la tabla no tiene una clave principal, pero la propiedad del marcador de trabajo está habilitada, debe proporcionar claves de marcadores de trabajo personalizadas. De lo contrario, la búsqueda de claves principales que se utilizarán como valor predeterminado fallará y la ejecución del trabajo fallará.
- Job bookmark keys sorting order (Orden de clasificación de claves de marcadores de trabajo): elija si los valores clave están en aumento o disminución secuencial.

Spark

- Schema (Esquema): ya que AWS Glue Studio utiliza la información almacenada en la conexión a fin de tener acceso al origen de datos en lugar de recuperar información de los metadatos de una tabla del Catálogo de datos, debe proporcionar los metadatos del esquema para el origen de datos. Elija Add schema (Agregar esquema) para abrir el editor de esquemas.

  Para obtener instrucciones sobre cómo utilizar el editor de esquemas, consulte Edición de esquema para un nodo de transformación personalizado.

- Connection options (Opciones de conexión): ingrese pares clave-valor adicionales según sea necesario para proporcionar información u opciones de conexión adicionales. Por ejemplo, puede ingresar un nombre de base de datos, un nombre de tabla, un nombre de usuario y una contraseña.

  Por ejemplo OpenSearch, para introducir los siguientes pares clave-valor, tal y como se describe en: the section called " Tutorial: uso de AWS Glue Connector for Elasticsearch "

  - `es.net.http.auth.user` : *`username`*
  - `es.net.http.auth.pass` : *`password`*
  - `es.nodes` : `https://`*`<Elasticsearch endpoint>`*
  - `es.port` : `443`
  - `path`: *`<Elasticsearch resource>`*
  - `es.nodes.wan.only` : `true`

Para ver un ejemplo de las opciones de conexión mínimas que se deben utilizar, consulte el ejemplo del script de prueba MinimalSparkConnectorTest.scala on GitHub, que muestra las opciones de conexión que normalmente proporcionaría en una conexión.

Athena

- Table name (Nombre de la tabla): el nombre de la tabla en el origen de datos. Si utilizas un conector para leer los CloudWatch registros de Athena, debes introducir el nombre de la tabla. `all_log_streams`
- Athena schema name (Nombre del esquema Athena): elija el esquema de su origen de datos Athena que corresponde a la base de datos que contiene la tabla. Si utiliza un conector para leer los CloudWatch registros de Athena, debe introducir un nombre de esquema similar a. `/aws/glue/`*`name`*
- Schema (Esquema): ya que AWS Glue Studio utiliza la información almacenada en la conexión para tener acceso al origen de datos en lugar de recuperar información de los metadatos de una tabla del Catálogo de datos, debe proporcionar los metadatos del esquema para el origen de datos. Elija Add schema (Agregar esquema) para abrir el editor de esquemas.

  Para obtener instrucciones sobre cómo utilizar el editor de esquemas, consulte Edición de esquema para un nodo de transformación personalizado.

- Additional connection options (Opciones adicionales de conexión): ingrese pares clave-valor adicionales según sea necesario para proporcionar información u opciones de conexión adicionales.

  Para ver un ejemplo, consulte el `README.md` archivo en https://github.com/aws-samples/aws-glue-samples /tree/master/ /Development/Athena GlueCustomConnectors. En los pasos de este documento, el código de muestra muestra las opciones de conexión mínimas necesarias, que son `tableName`, `schemaName` y `className`. En el ejemplo del código se especifican estas opciones como parte de la variable `optionsMap`, pero puede especificarlos para su conexión y luego usar la conexión.

3. (Opcional) después de proporcionar la información necesaria, puede ver el esquema de datos resultante para su origen de datos al seleccionar la pestaña Output schema (Esquema de salida) en el panel de detalles del nodo. Los nodos secundarios que agregue al gráfico de trabajo utilizan el esquema que se muestra en esta pestaña.
4. (Opcional) después de configurar las propiedades del nodo y del origen de datos, puede ver la previsualización del conjunto de datos para su origen de datos al seleccionar la pestaña Data preview (Previsualización de datos) en el panel de detalles del nodo. La primera vez que elija esta pestaña para cualquier nodo de trabajo, se le pedirá que proporcione un rol de IAM para acceder a los datos. Hay un costo asociado con el uso de este recurso y la facturación comienza tan pronto como proporcione un rol de IAM.

Configuración de las propiedades de destino para los nodos que utilizan conectores

Si utiliza un conector para el tipo de destino de datos, debe configurar las propiedades del nodo de destino de datos.

Para configurar las propiedades de un nodo de destino de datos que utiliza un conector

1. Elija el nodo de destino de datos para el conector en el gráfico de trabajo. A continuación, en el lado derecho, en el panel de detalles del nodo, seleccione la pestaña Data target properties (Propiedades de destino de datos), si aún no está seleccionada.
2. En la pestaña Data target properties (Propiedades de Destino de datos), elija la conexión que se utilizará para escribir en el destino.

   Ingrese la información adicional necesaria para cada tipo de conexión:

JDBC

- Connection (Conexión): elija la conexión que desea utilizar con el conector. Para obtener información acerca de cómo crear una conexión, consulte Creación de conexiones para conectores.
- Table name (Nombre de la tabla): el nombre de la tabla en el destino de datos. Si el destino de datos no utiliza la tabla de términos, proporcione el nombre de la estructura de datos adecuada, tal y como se indica en la información de uso del conector personalizado (que está disponible en). AWS Marketplace
- Batch size (Tamaño del lote) (opcional): ingrese el número de filas o registros que desea insertar en la tabla de destino en una sola operación. El valor predeterminado es 1000 filas.

Spark

- Connection (Conexión): elija la conexión que desea utilizar con el conector. Si no creó una conexión anteriormente, elija Create connection (Crear conexión) para crear una. Para obtener información acerca de cómo crear una conexión, consulte Creación de conexiones para conectores.
- Connection options (Opciones de conexión): ingrese pares clave-valor adicionales según sea necesario para proporcionar información u opciones de conexión adicionales. Puede ingresar un nombre de base de datos, un nombre de tabla, un nombre de usuario y una contraseña.

  Por ejemplo OpenSearch, para introducir los siguientes pares clave-valor, tal y como se describe en: the section called “ Tutorial: uso de AWS Glue Connector for Elasticsearch ”

  - `es.net.http.auth.user` : *`username`*
  - `es.net.http.auth.pass` : *`password`*
  - `es.nodes` : `https://`*`<Elasticsearch endpoint>`*
  - `es.port` : `443`
  - `path`: *`<Elasticsearch resource>`*
  - `es.nodes.wan.only` : `true`

Para ver un ejemplo de las opciones de conexión mínimas que se deben utilizar, consulte el ejemplo del script de prueba MinimalSparkConnectorTest.scala on GitHub, que muestra las opciones de conexión que normalmente proporcionaría en una conexión.

3. Después de proporcionar la información necesaria, puede ver el esquema de datos resultante para su origen de datos al seleccionar la pestaña Output schema (Esquema de salida) en el panel de detalles del nodo.

Administración de conectores y conexiones

Utilice la página Conexiones en AWS Glue para administrar los conectores y las conexiones.

Temas

- Visualización de detalles del conector y la conexión
- Edición de conectores y conexiones
- Eliminación de conectores y conexiones
- Cancelar una suscripción para un conector

Visualización de detalles del conector y la conexión

Puede ver información resumida acerca de los conectores y las conexiones en las tablas de recursos Your connectors (Sus conectores) y Your connections (Sus conexiones) en la página Connectors (Conectores). Para ver la información detallada, siga estos pasos.

Para ver los detalles del conector o la conexión

1. En la consola de AWS Glue Studio, elija Connectors (Conectores) en el panel de navegación de la consola.
2. Elija el conector o la conexión para el/la que desea ver información detallada.
3. Seleccione Actions (Acciones) y luego elija View details (Ver detalles) para abrir la página de detalles de ese conector o conexión.
4. En la página de detalles, puede elegir Editar o Eliminar el conector o la conexión.
   - En el caso de los conectores, puede elegir Create connection (Crear conexión) para crear una conexión nueva que utilice el conector.

- En el caso de las conexiones, puede elegir Create job (Crear trabajo) para crear un trabajo que utilice la conexión.

Edición de conectores y conexiones

Se utiliza la página Connectors (Conectores) para modificar la información almacenada en sus conectores y conexiones.

Para modificar un conector o una conexión

1. En la consola de AWS Glue Studio, elija Connectors (Conectores) en el panel de navegación de la consola.
2. Elija el conector o conexión que desea modificar.
3. Seleccione Acciones y, a continuación, Editar.

   También puede elegir View details (Ver detalles) y en la página de detalles del conector o la conexión, puede elegir Edit (Editar).
4. En la página Edit connector (Editar conector) o Edit connection (Editar conexión), actualice la información y, a continuación, elija Save (Guardar).

Eliminación de conectores y conexiones

Se utiliza la página Connectors (Conectores) para eliminar conectores y conexiones. Si elimina un conector, también se deben eliminar todas las conexiones que se hayan creado para ese conector.

Para quitar conectores de AWS Glue Studio

1. En la consola de AWS Glue Studio, elija Connectors (Conectores) en el panel de navegación de la consola.
2. Elija el conector o conexión que desea eliminar.
3. Elija Acciones y, a continuación, elija Eliminar.

   También puede elegir View details (Ver detalles) y en la página de detalles del conector o la conexión, puede elegir Delete (Eliminar).
4. Compruebe que desea eliminar el conector o la conexión mediante la introducción de **`Delete`** y luego elija Delete (Eliminar).

   Al eliminar un conector, también se eliminan todas las conexiones creadas para ese conector.

Los trabajos que utilicen una conexión eliminada ya no funcionarán. Puede editar los trabajos para que utilicen un almacén de datos diferente o eliminar los trabajos. Para obtener información sobre cómo eliminar un trabajo, consulte Eliminación de trabajos.

Si elimina un conector, esto no cancela la suscripción del conector en AWS Marketplace. Para cancelar una suscripción de un conector eliminado, siga las instrucciones en Cancelar una suscripción para un conector.

Cancelar una suscripción para un conector

Tras eliminar las conexiones y el conectorAWS Glue Studio, puedes cancelar tu suscripción AWS Marketplace si ya no necesitas el conector.

> **Note**
>
> Si cancela su suscripción a un conector, esto no elimina el conector o la conexión de su cuenta. Cualquier trabajo que utilice el conector y las conexiones relacionadas ya no podrá usar el conector y fallará.
> Antes de cancelar la suscripción o volver a suscribirse a un conector de AWS Marketplace, debe eliminar las conexiones y los conectores existentes asociados a ese AWS Marketplace producto.

Para cancelar la suscripción a un conector en AWS Marketplace

1. Inicie sesión en la AWS Marketplace consola en https://console.aws.amazon.com/marketplace.
2. Elija Manage subscriptions (Administrar suscripciones).
3. En la página Manage subscriptions (Administrar suscripciones), elija Manage (Administrar) junto a la suscripción del conector que desea cancelar.
4. Elija Actions (Acciones) y después Cancel Subscription (Cancelar suscripción).
5. Seleccione la casilla de verificación para confirmar que las instancias en ejecución se cargan a su cuenta y, a continuación, elija Yes, cancel subscription (Sí, cancelar suscripción).

Desarrollo de conectores personalizados

Puede escribir el código que lee datos del almacén de datos o escribe datos en él y los formatea para utilizarlos con trabajos de AWS Glue Studio. Puede crear conectores para los almacenes

de datos de Spark, Athena y JDBC. El código de muestra publicado en GitHub proporciona una descripción general de las interfaces básicas que debe implementar.

Será necesario un entorno de desarrollo local para crear su código de conector. Puede usar cualquier IDE o incluso solo un editor de línea de comandos para escribir su conector. Algunos ejemplos de entornos de desarrollo incluyen los siguientes:

- Un entorno Scala local con una biblioteca AWS Glue ETL Maven local, como se describe en Desarrollo local con Scala en la Guía para desarrolladores de AWS Glue .
- IDE de IntelliJ, mediante la descarga de IDE desde https://www.jetbrains.com/idea/.

Temas

- Desarrollo de conectores Spark
- Desarrollo de conectores Athena
- Desarrollo de conectores JDBC
- Ejemplos de uso de conectores personalizados con AWS Glue Studio
- Desarrollo de conectores para AWS GlueAWS Marketplace

Desarrollo de conectores Spark

Puedes crear un conector Spark con la DataSource API V2 de Spark (Spark 2.4) para leer los datos.

Para crear un conector Spark personalizado

Sigue los pasos de la biblioteca de AWS Glue GitHub ejemplos para desarrollar conectores Spark, que se encuentra en https://github.com/aws-samples/ aws-glue-samples /tree/master/ /development/ spark/README.md GlueCustomConnectors.

Desarrollo de conectores Athena

Puede crear un conector de Athena para que lo utilice AWS Glue y AWS Glue Studio con la finalidad de consultar un origen de datos personalizado.

Para crear un conector Athena personalizado

Siga los pasos de la biblioteca de AWS Glue GitHub ejemplos para desarrollar los conectores Athena, que se encuentra en https://github.com/aws-samples/ aws-glue-samples /tree/master/ GlueCustomConnectors /development/Athena.

Desarrollo de conectores JDBC

Puede crear un conector que utilice JDBC para acceder a los almacenes de datos.

Para crear un conector JDBC personalizado

1. Instale las bibliotecas de tiempo de ejecución Spark de AWS Glue en su entorno de desarrollo local. Consulte las instrucciones de la biblioteca de ejemplos en https://github.com/aws-samples/ /tree/master/ /development/ /README.md. AWS Glue GitHub aws-glue-samples GlueCustomConnectors GlueSparkRuntime
2. Implemente el controlador JDBC que es responsable de recuperar los datos del origen de datos. Consulte la documentación de Java para Java SE 8.

   Cree un punto de entrada dentro del código que AWS Glue Studio utiliza para localizar el conector. El campo Class name (Nombre de clase) debe ser la ruta completa de su controlador JDBC.
3. Use la API `GlueContext` para leer datos con el conector. Los usuarios pueden agregar más opciones de entrada en la consola de AWS Glue Studio para configurar la conexión con el origen de datos, si es necesario. Para ver un ejemplo de código que muestra cómo leer y escribir en una base de datos JDBC con un conector JDBC personalizado, consulte Valores personalizados y ConnectionType. AWS Marketplace

Ejemplos de uso de conectores personalizados con AWS Glue Studio

Puede consultar los siguientes blogs para ver ejemplos de uso de conectores personalizados:

- Desarrollar, probar e implementar conectores personalizados para los almacenes de datos con AWS Glue
- Apache Hudi: Writing to Apache Hudi tables using AWS Glue Custom Connector (Escribir en tablas Apache Hudi con conectores personalizados de Glue)
- Google BigQuery: migración de datos de Google BigQuery a Amazon S3 mediante conectores AWS Glue personalizados
- Snowflake (JDBC): Performing data transformations using Snowflake and AWS Glue (Realizar transformaciones de datos con Snowflake y Glue)
- SingleStore: Construir un ETL rápido usando SingleStore y AWS Glue

- Salesforce: Ingest Salesforce data into Amazon S3 using the CData JDBC custom connector with AWS Glue(Capturar datos de Salesforce en Amazon S3 mediante el conector personalizado CData JDBC con AWS Glue)
- MongoDB: Building AWS Glue Spark ETL jobs using Amazon DocumentDB (with MongoDB compatibility) and MongoDB (Crear trabajos de ETL de Glue Spark mediante Amazon DocumentDB [compatible con MongoDB] y MongoDB)
- Amazon Relational Database Service (Amazon RDS): Cree trabajos de ETL en AWS Glue Spark incorporando sus propios controladores JDBC para Amazon RDS
- MySQL (JDBC): https://github.com/aws-samples/ aws-glue-samples /blob/master/ GlueCustomConnectors /Development/Spark/ SQL.scala SparkConnectorMy

Desarrollo de conectores para AWS GlueAWS Marketplace

Como AWS socio, puede crear conectores personalizados y subirlos para AWS Marketplace venderlos a AWS Glue los clientes.

El proceso para desarrollar el código del conector es el mismo que para los conectores personalizados, pero el proceso de carga y verificación del código del conector es más detallado. Consulte las instrucciones del GitHub sitio web AWS Marketplace sobre cómo crear conectores.

Restricciones para el uso de conectores y conexiones en AWS Glue Studio

Cuando utilices conectores personalizados o conectores de AWS Marketplace, ten en cuenta las siguientes restricciones:

- La API TestConnection no se admite con conexiones creadas para conectores personalizados.
- El cifrado de las contraseñas de conexión del Catálogo de datos no se admite con conectores personalizados.
- No puede utilizar marcadores de trabajo si especifica un predicado de filtro para un nodo de origen de datos que utiliza un conector JDBC.
- No se admite la creación de una conexión a Marketplace fuera de la interfaz de AWS Glue Studio usuario.

Conexión con orígenes de datos mediante trabajos de ETL visuales

Al crear un nuevo trabajo, puede usar conexiones para conectarse a datos cuando edite trabajos de ETL visuales en AWS Glue. Para ello, puede agregar nodos de origen que usen conectores para leer datos y nodos de destino para especificar la ubicación para escribir datos.

Temas

- Modificar las propiedades de un nodo de origen de datos
- Uso de tablas del Catálogo de datos para el origen de datos
- Uso de un conector para el origen de datos
- Uso de archivos en Amazon S3 para el origen de datos
- Uso de un origen de datos de streaming
- Referencias

Modificar las propiedades de un nodo de origen de datos

Para especificar las propiedades del origen de datos, elija primero un nodo de origen de datos en el diagrama de trabajo. A continuación, en el lado derecho del panel de detalles del nodo, configure las propiedades del nodo.

Para modificar las propiedades de un nodo de origen de datos

1. Vaya al editor visual para acceder a un trabajo nuevo o guardado.
2. Elija un nodo de origen de datos en el diagrama de trabajo.
3. Elija la pestaña Node properties (Propiedades del nodo) en el panel de detalles del nodo y escriba la siguiente información:
   - Name (Nombre): (opcional) ingrese un nombre para asociar al nodo en el diagrama de trabajo. Este nombre debe ser único entre todos los nodos de este trabajo.
   - Node type (Tipo de nodo): el tipo de nodo determina la acción que realiza el nodo. En la lista de opciones para Node type (Tipo de nodo), elija uno de los valores enumerados en el encabezado Data source (Origen de datos).
4. Configuración de la información de Data source properties (Propiedades de origen de datos). Para obtener más información, consulte las siguientes secciones:
   - Uso de tablas del Catálogo de datos para el origen de datos

- Uso de un conector para el origen de datos
- Uso de archivos en Amazon S3 para el origen de datos
- Uso de un origen de datos de streaming

5. (Opcional) después de configurar las propiedades del nodo y del origen de datos, puede ver el esquema de datos para su origen de datos al seleccionar la pestaña Output schema (Esquema de salida) en el panel de detalles del nodo. La primera vez que elija esta pestaña para cualquier nodo de trabajo, se le pedirá que proporcione un rol de IAM para acceder a los datos. Si no ha especificado un rol de IAM en Job details (Detalles del trabajo), se le pedirá que ingrese un rol de IAM aquí.
6. (Opcional) después de configurar las propiedades del nodo y del origen de datos, puede ver la previsualización del conjunto de datos para su origen de datos al seleccionar la pestaña Data preview (Previsualización de datos) en el panel de detalles del nodo. La primera vez que elija esta pestaña para cualquier nodo de trabajo, se le pedirá que proporcione un rol de IAM para acceder a los datos. Hay un costo asociado con el uso de este recurso y la facturación comienza tan pronto como proporcione un rol de IAM.

Uso de tablas del Catálogo de datos para el origen de datos

En todos los orígenes de datos, excepto Amazon S3 y los conectores, debe existir una tabla en el AWS Glue Data Catalog para el tipo de fuente que elija. AWS Glue no crea la tabla del Catálogo de datos.

Para configurar un nodo de origen de datos en función de una tabla del Catálogo de datos

1. Vaya al editor visual para acceder a un trabajo nuevo o guardado.
2. Elija un nodo de origen de datos en el diagrama de trabajo.
3. Elija la pestaña Data source properties (Propiedades de origen de datos) y, a continuación, escriba la información siguiente:

   - S3 source type (Tipo de origen S3): (solo para orígenes de datos de Amazon S3) elija la opción Select a Catalog table (Seleccionar una tabla del catálogo) para utilizar una tabla del AWS Glue Data Catalog existente.
   - Database (Base de datos): elija la base de datos del Catálogo de datos que contiene la tabla de origen que desea utilizar para este trabajo. Puede utilizar el campo de búsqueda para buscar una base de datos por su nombre.

- Table (Tabla): elija la tabla asociada a los datos de origen de la lista. Esta tabla ya debe existir en AWS Glue Data Catalog. Puede utilizar el campo de búsqueda para buscar una tabla por su nombre.
- Partition predicate (Predicado de partición): (sólo para orígenes de datos de Amazon S3) ingrese una expresión booleana basada en Spark SQL que incluya sólo las columnas de partición. Por ejemplo: `"(year=='2020' and month=='04')"`.
- Temporary directory (Directorio temporal): (sólo para orígenes de datos de Amazon Redshift) ingrese una ruta de acceso para la ubicación de un directorio de trabajo en Amazon S3 donde su trabajo de ETL puede escribir resultados intermedios temporales.
- Role associated with the cluster (Rol asociado al clúster): (sólo para orígenes de datos de Amazon Redshift) ingrese un rol para que utilice su trabajo de ETL que contenga permisos para clústeres de Amazon Redshift. Para obtener más información, consulte the section called “Permisos de origen de datos y destino de datos”.

Uso de un conector para el origen de datos

Si selecciona un conector para el Node type (Tipo de nodo), siga las instrucciones en Creación de trabajos con conectores personalizados para finalizar la configuración de las propiedades del origen de datos.

Uso de archivos en Amazon S3 para el origen de datos

Si elige Amazon S3 como origen de datos, puede elegir entre lo siguiente:

- Una base de datos y una tabla del Catálogo de datos.
- Un bucket, carpeta o archivo en Amazon S3.

Si utiliza un bucket de Amazon S3 como origen de datos, AWS Glue detecta el esquema de los datos en la ubicación especificada desde uno de los archivos o al utilizar el archivo especificado como archivo de muestra. La detección de esquemas se produce cuando se utiliza el botón Infer schema (Inferir esquema). Si cambia la ubicación de Amazon S3 o el archivo de ejemplo, debe elegir Infer schema (Inferir esquema) una vez más, para realizar la detección de esquemas utilizando la nueva información.

Para configurar un nodo de origen de datos que lea directamente desde archivos en Amazon S3

1. Vaya al editor visual para acceder a un trabajo nuevo o guardado.

2. Elija un nodo de origen de datos en el diagrama de trabajo para un origen de Amazon S3.
3. Elija la pestaña Data source properties (Propiedades de origen de datos) y, a continuación, escriba la información siguiente:

   - S3 source type (Tipo de origen de S3): (solo para orígenes de datos de Amazon S3) elija la opción S3 location (Ubicación de S3).
   - S3 URL (URL de S3): ingrese la ruta de acceso al bucket, carpeta o archivo de Amazon S3 que contiene los datos de su trabajo. Puede elegir Browse S3 (Examinar S3) para seleccionar la ruta de acceso entre las ubicaciones disponibles para su cuenta.
   - Recursive (Acción recursiva): elija esta opción si desea que AWS Glue lea datos de archivos en carpetas secundarias en la ubicación de S3.

     Si las carpetas secundarias contienen datos particionados, AWS Glue no agrega ninguna información de partición especificada en los nombres de carpeta al Catálogo de datos. Por ejemplo, considere la siguientes carpetas en Amazon S3:

     ```
     S3://sales/year=2019/month=Jan/day=1
     S3://sales/year=2019/month=Jan/day=2
     ```

     Si elige Recursive (Acción recursiva) y selecciona la carpeta `sales` como la ubicación de S3, AWS Glue lee los datos en todas las carpetas secundarias, pero no crea particiones por año, mes o día.
   - Data format (Formato de datos): elija el formato en el que se almacenan los datos. Puede elegir JSON, CSV o Parquet. El valor que seleccione indica al trabajo de AWS Glue cómo leer los datos del archivo de origen.

     > **Note**
     >
     > Si no selecciona el formato correcto para los datos, AWS Glue puede inferir bien el esquema, pero el trabajo no podrá analizar los datos del archivo de origen en forma correcta.

     Puede introducir opciones de configuración adicionales, en función del formato que elija.

     - JSON (JavaScript Object Notation)

- JsonPath: ingrese una ruta de JSON que apunte a un objeto que se usa para definir un esquema de tabla. Las expresiones de la ruta JSON siempre hacen referencia a una estructura JSON de la misma manera que la expresión XPath se utiliza en combinación con un documento XML. El “root member object (objeto miembro raíz)” en la ruta JSON siempre se conoce como $, incluso si es un objeto o matriz. La ruta de JSON se puede escribir en la notación de puntos o la notación de corchete.

  Para obtener más información acerca de las rutas JSON, consulte JsonPath en el sitio web GitHub.

- Records in source files can span multiple lines (Los registros en los archivos de origen pueden abarcar varias líneas): elija esta opción si un solo registro puede abarcar varias líneas en el archivo CSV.

- CSV (valores separados por comas)
  - Delimiter (Delimitador): escriba un carácter para indicar qué elemento va a separar cada entrada de columna en la fila, por ejemplo, `;` o `,`.
  - Escape character (Carácter de escape): escriba un carácter que se utilice como carácter de escape. Este carácter indica que el carácter que sigue inmediatamente al carácter de escape debe tomarse en forma literal y no debe interpretarse como un delimitador.
  - Quote characters (Caracteres de cita): ingrese el carácter que se utiliza para agrupar cadenas separadas en un solo valor. Por ejemplo, debería elegir Double quote (") [Comilla doble (“)] si tiene valores como `"This is a single value"` en su archivo CSV.
  - Records in source files can span multiple lines (Los registros en los archivos de origen pueden abarcar varias líneas): elija esta opción si un solo registro puede abarcar varias líneas en el archivo CSV.
  - First line of source file contains column headers (La primera línea del archivo de origen contiene encabezados de columna): elija esta opción si la primera fila del archivo CSV contiene encabezados de columna en lugar de datos.
- Parquet (almacenamiento en columna de Apache Parquet)

  No hay ajustes adicionales que configurar para los datos almacenados en formato Parquet.

- Partition predicate (Predicado de partición): para particionar los datos que se leen desde el origen de datos, ingrese una expresión booleana basada en Spark SQL que incluya sólo las columnas de partición. Por ejemplo: `"(year=='2020' and month=='04')"`.

- Advanced options (Opciones avanzadas): expanda esta sección si desea que AWS Glue detecte el esquema de los datos en función de un archivo específico.
  - Schema inference (Inferencia de esquema): elija la opción Choose a sample file from S3 (Elegir un archivo de ejemplo desde S3) si desea utilizar un archivo específico en lugar de permitir que AWS Glue elija un archivo.
  - Auto-sampled file (Archivo de ejemplo automático): ingrese la ruta de acceso al archivo en Amazon S3 que se utilizará para inferir el esquema.

  Si está editando un nodo de origen de datos y cambia el archivo de ejemplo seleccionado, elija Reload schema (Volver a cargar esquema) para detectar el esquema mediante el nuevo archivo de ejemplo.

4. Elija el botón Infer schema (Inferir esquema) para detectar el esquema a partir de los archivos de origen en Amazon S3. Si cambia la ubicación de Amazon S3 o el archivo de ejemplo, debe elegir Infer schema (Inferir esquema) una vez más, para inferir el esquema con la nueva información.

Uso de un origen de datos de streaming

Puede crear trabajos de extracción, transformación y carga (ETL) de streaming que se ejecuten en forma continua y consuman datos de orígenes de streaming en Amazon Kinesis Data Streams, Apache Kafka y Amazon Managed Streaming for Apache Kafka (Amazon MSK).

Para configurar las propiedades de un origen de datos de streaming

1. Vaya al editor visual de gráficos para acceder a un trabajo nuevo o guardado.
2. Elija un nodo de origen de datos en el gráfico para Kafka o Kinesis Data Streams.
3. Elija la pestaña Data source properties (Propiedades de origen de datos) y, a continuación, escriba la información siguiente:

   Kinesis

   - Kinesis source type (Tipo de origen de Kinesis): elija la opción Stream details (Detalles de la transmisión) para utilizar el acceso directo a la fuente de streaming o elegir Data Catalog table (Tabla del Catálogo de datos) a fin de utilizar la información almacenada allí en su lugar.

     Si elige Stream details (Detalles de la transmisión), especifique la siguiente información adicional.

- Ubicación del flujo de datos: elija si el flujo se encuentra asociado al usuario actual o si se encuentra asociado a otro usuario.
- Region (Región): elija la Región de AWS donde existe la transmisión. Esta información se utiliza con la finalidad de crear el ARN para acceder a la secuencia de datos.
- Stream ARN (Transmisión de ARN): ingrese el nombre de recurso de Amazon (ARN) para Kinesis Data Stream. Si la transmisión se encuentra en la cuenta corriente, puede elegir el nombre de la secuencia en la lista desplegable. Puede utilizar el campo de búsqueda para buscar una secuencia de datos de datos por su nombre o ARN.
- Data format (Formato de los datos): elija el formato utilizado por la secuencia de datos de la lista.

  AWS Glue detecta de forma automática el esquema de los datos de streaming.

Si elige Data Catalog table (Tabla del Catálogo de datos), especifique la siguiente información adicional.

- Database (Base de datos): (opcional) elija la base de datos en el Catálogo de datos de AWS Glue que contenga la tabla asociada al origen de datos de streaming. Puede utilizar el campo de búsqueda para buscar una base de datos por su nombre.
- Table (Tabla): (opcional) elija la tabla asociada a los datos de origen de la lista. Esta tabla ya debe existir en el Catálogo de datos de AWS Glue. Puede utilizar el campo de búsqueda para buscar una tabla por su nombre.
- Detect schema (Detectar esquemas): elija esta opción para que AWS Glue detecte el esquema a partir de los datos de streaming, en lugar de almacenar la información del esquema en una tabla del Catálogo de datos. Esta opción se habilita automáticamente si elige la opción Stream details (Detalles de la transmisión).

- Starting position (Posición inicial): de forma predeterminada, el trabajo de ETL utiliza la opción Earliest (La primera), lo que significa que lee los datos desde el registro más antiguo disponible en la transmisión. En su lugar, puede elegir Latest (El último), lo que indica que el trabajo de ETL debería empezar a leer justo después del registro más reciente en la transmisión.
- Window size (Tamaño de ventana): de forma predeterminada, su trabajo de ETL procesa y escribe datos en ventanas de 100 segundos. Esto permite que los datos se procesen de forma eficiente y permite que las agregaciones se realicen en los datos que llegan más tarde de lo previsto. Puede modificar este tamaño de ventana para aumentar la puntualidad o la precisión de agregación.

Los trabajos de streaming de AWS Glue utilizan puntos de control en lugar de marcadores de trabajo para realizar un seguimiento de los datos leídos.

- Advanced connection options (Opciones avanzadas de conexión): expanda esta sección para agregar pares de valor de clave a fin de especificar opciones de conexión adicionales. Para obtener más información acerca de las opciones que puede especificar aquí, consulte "connectionType": "kinesis" en la Guía para desarrolladores de AWS Glue.

Kafka

- Apache Kafka source (Fuente de Apache Kafka): elija la opción Stream details (Detalles de la transmisión) para utilizar el acceso directo a la fuente de streaming o elegir Data Catalog table (Tabla del Catálogo de datos) a fin de utilizar la información almacenada allí en su lugar.

  Si elige Data Catalog table (Tabla del Catálogo de datos), especifique la siguiente información adicional.

  - Database (Base de datos): (opcional) elija la base de datos en el Catálogo de datos de AWS Glue que contenga la tabla asociada al origen de datos de streaming. Puede utilizar el campo de búsqueda para buscar una base de datos por su nombre.
  - Table (Tabla): (opcional) elija la tabla asociada a los datos de origen de la lista. Esta tabla ya debe existir en el Catálogo de datos de AWS Glue. Puede utilizar el campo de búsqueda para buscar una tabla por su nombre.
  - Detect schema (Detectar esquemas): elija esta opción para que AWS Glue detecte el esquema a partir de los datos de streaming, en lugar de almacenar la información del esquema en una tabla del Catálogo de datos. Esta opción se habilita automáticamente si elige la opción Stream details (Detalles de la transmisión).

  Si elige Stream details (Detalles de la transmisión), especifique la siguiente información adicional.

  - Connection name (Nombre de la conexión): elija la conexión de AWS Glue que contiene la información de acceso y autenticación para la secuencia de datos de Kafka. Debe utilizar una conexión con los orígenes de datos de streaming de Kafka. Si no existe una conexión, puede utilizar la consola de AWS Glue a fin de crear una conexión para la secuencia de datos de Kafka.
  - Topic name (Nombre del tema): ingrese el nombre del tema en el que se va a leer.

  - Data format (Formato de los datos): elija el formato que desea utilizar al leer datos de la secuencia de eventos de Kafka.
- Starting position (Posición inicial): de forma predeterminada, el trabajo de ETL utiliza la opción Earliest (La primera), lo que significa que lee los datos desde el registro más antiguo disponible en la transmisión. En su lugar, puede elegir Latest (El último), lo que indica que el trabajo de ETL debería empezar a leer justo después del registro más reciente en la transmisión.
- Window size (Tamaño de ventana): de forma predeterminada, su trabajo de ETL procesa y escribe datos en ventanas de 100 segundos. Esto permite que los datos se procesen de forma eficiente y permite que las agregaciones se realicen en los datos que llegan más tarde de lo previsto. Puede modificar este tamaño de ventana para aumentar la puntualidad o la precisión de agregación.

  Los trabajos de streaming de AWS Glue utilizan puntos de control en lugar de marcadores de trabajo para realizar un seguimiento de los datos leídos.
- Advanced connection options (Opciones avanzadas de conexión): expanda esta sección para agregar pares de valor de clave a fin de especificar opciones de conexión adicionales. Para obtener más información acerca de las opciones que puede especificar aquí, consulte "connectionType": "kafka" en la Guía para desarrolladores de AWS Glue.

> **Note**
>
> Las previsualizaciones de datos no se soportan actualmente para los orígenes de datos de streaming.

Referencias

prácticas recomendadas

- Cree una canalización de servicios de ETL para cargar datos de forma incremental desde Amazon S3 hasta utilizar Amazon Redshift en AWS Glue

Programación de ETL

- Tipos de conexión y opciones para ETL en AWS Glue

- Valores connectionType de JDBC
- Opciones avanzadas para mover datos hacia y desde Amazon Redshift

Agregar una conexión JDBC con sus propios controladores JDBC

Puede utilizar su propio controlador JDBC cuando utilice una conexión JDBC. Si el controlador predeterminado utilizado por el rastreador AWS Glue no puede conectarse a una base de datos, puede utilizar su propio controlador JDBC. Por ejemplo, si desea utilizar el SHA-256 con su base de datos de Postgres y los controladores Postgres más antiguos no lo admiten, puede utilizar su propio controlador JDBC.

Orígenes de datos compatibles

Orígenes de datos compatibles	Orígenes de datos no compatibles
MySQL	Snowflake
Postgres	
Oracle	
Redshift	
SQL Server	
Aurora*	

*Se admite si se utiliza el controlador JDBC nativo. No se pueden aprovechar todas las funciones del controlador.

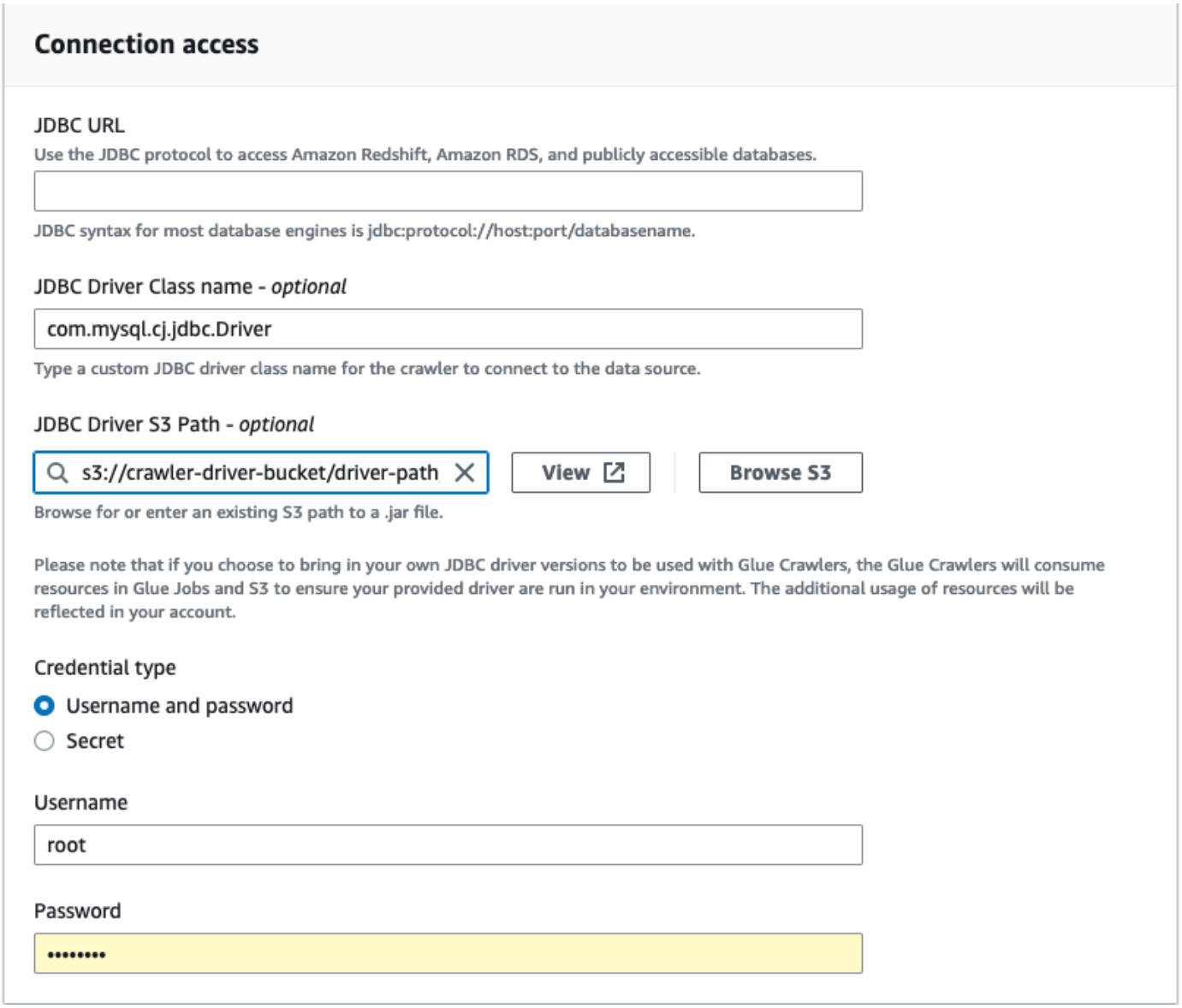
Agregar un controlador JDBC a una conexión JDBC

> **Note**
>
> Si decide incorporar sus versiones de controladores JDBC, los rastreadores AWS Glue consumirán recursos en trabajos AWS Glue y en los buckets de Amazon S3 para garantizar que los controladores proporcionados se ejecuten en su entorno. El uso adicional de los recursos se reflejará en su cuenta. El costo de los rastreadores AWS Glue y los trabajos

se incluye en la categoría AWS Glue de facturación. Además, proporcionar su propio controlador JDBC no significa que el rastreador pueda aprovechar todas las funciones del controlador.

Agregar su propio controlador JDBC a una conexión JDBC:

1. Agregue el archivo del controlador JDBC a una ubicación de Amazon S3. Puede crear un bucket o una carpeta o utilizar un bucket o una carpeta existente.
2. En la consola AWS Glue, seleccione Conexiones en el menú de la izquierda, en el catálogo de datos y luego cree una conexión nueva.
3. Rellene los campos de las Propiedades de la conexión y elija JDBC para Tipo de conexión.
4. En Acceso a la conexión, ingrese la URL del JDBC y el nombre de la clase de controlador del JDBC: opcional. El nombre de la clase de controlador debe corresponder a un origen de datos compatible con los rastreadores AWS Glue.

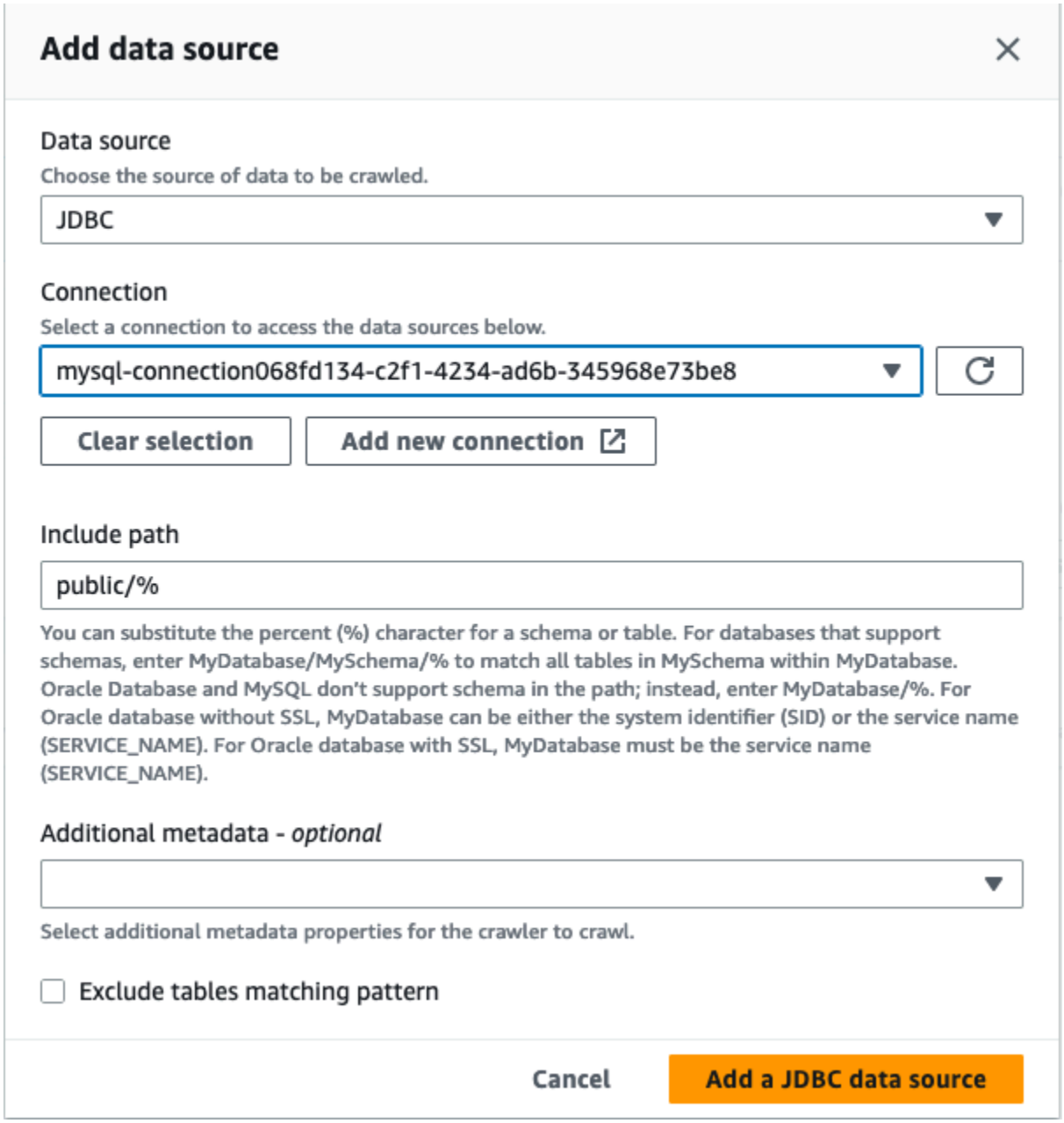


5. Elija la ruta de Amazon S3 en la que se encuentra el controlador JDBC en el campo Ruta Amazon S3 del controlador JDBC: opcional.
6. Rellene los campos correspondientes al tipo de credencial si ingresa un nombre de usuario y una contraseña o un secreto. Cuando haya terminado, elija Crear conexión.

> **Note**
>
> Las pruebas de conexiones no son compatibles en la actualidad. Al rastrear el origen de datos con un controlador JDBC que haya proporcionado, el rastreador omite este paso.

7. Agregue la conexión recién creada a un rastreador. En la consola AWS Glue, seleccione Rastreadores en el menú de la izquierda, en el Catálogo de datos y cree un rastreador nuevo.

8. En el asistente Agregar rastreadores, en el paso 2, elija Agregar un origen de datos.



9. Elija JDBC como origen de datos y elija la conexión que se creó en los pasos anteriores. Completado
10. Para utilizar su propio controlador JDBC con un AWS Glue rastreador, añada los siguientes permisos a la función utilizada por el rastreador:

    - Conceda permisos para las siguientes acciones de trabajos: `CreateJob`, `DeleteJob`, `GetJob`, `GetJobRun`, `StartJobRun`.
    - Conceda permisos para las acciones de IAM: `iam:PassRole`

- Conceda permisos para todas las acciones de Amazon S3: `s3:DeleteObjects`, `s3:GetObject`, `s3:ListBucket`, `s3:PutObject`.
- Conceda al director de servicio acceso al bucket o carpeta en la política de IAM.

Política de IAM de ejemplo:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::bucket-name/driver-parent-folder/driver.jar",
                "arn:aws:s3:::bucket-name"
            ]
        }
    ]
}
```

11. Si utiliza una VPC, debe permitir el acceso al punto de conexión AWS Glue al crear el punto de conexión de la interfaz y agregarlo a la tabla de enrutamiento. Para obtener más información, consulte Creación de un punto de conexión de VPC para AWS Glue.
12. Si utiliza el cifrado en su catálogo de datos, cree el punto final de la AWS KMS interfaz y agréguelo a la tabla de enrutamiento. Para obtener más información, consulte Crear un punto de conexión de VPC para AWS KMS.

Prueba de una conexión de AWS Glue

Como práctica recomendada, antes de utilizar una conexión a AWS Glue en un trabajo de ETL, utilice la consola de AWS Glue para probar la conexión. AWS Glue utiliza los parámetros de la

conexión para confirmar que puede obtener acceso al almacén de datos e informa de cualquier error. Para obtener más información acerca de las conexiones a AWS Glue, consulte Conexión a datos.

Para probar una conexión a AWS Glue

1. Inicie sesión en la AWS Glue consola AWS Management Console y ábrala en https://console.aws.amazon.com/glue/.
2. En el panel de navegación, en Data Catalog, elija Conexiones. También puede elegir Conexiones de datos sobre el Data Catalog en el panel de navegación.
3. En Conexiones, seleccione la casilla de verificación situada junto a la conexión deseada y, a continuación, elija Acciones. En el menú desplegable, elija Probar conexión.
4. En el cuadro de diálogo Probar conexión, seleccione un rol o elija Crear rol de IAM para ir a la consola AWS Identity and Access Management (IAM) y crear un nuevo rol. El rol debe tener permisos en el almacén de datos.
5. Elija Confirmar.

   La prueba comienza y puede tardar varios minutos en completarse. Si la prueba no se realiza correctamente, seleccione Solucionar problemas para ver los pasos necesarios para resolver el problema.
6. Seleccione Registros para ver los inicios de sesión. CloudWatch Debe tener los permisos de IAM necesarios para ver los registros. Para obtener más información, consulte Políticas AWS administradas (predefinidas) para CloudWatch registros en la Guía del usuario de Amazon CloudWatch Logs.

Configuración de las llamadas de AWS para que pasen a través de la VPC

El parámetro de trabajo especial `disable-proxy-v2` permite enrutar las llamadas a servicios tales como Amazon S3, CloudWatch y AWS Glue a través de la VPC. De forma predeterminada, AWS Glue usa un proxy local para enviar tráfico a través de la VPC de AWS Glue a fin de descargar scripts y bibliotecas de Amazon S3, enviar solicitudes a CloudWatch para publicar registros y métricas y enviar solicitudes a AWS Glue para acceder a catálogos de datos. Este proxy permite que el trabajo funcione con normalidad, aunque su VPC no configure una ruta adecuada a otros servicios de AWS, como Amazon S3, CloudWatch y AWS Glue. Ahora, AWS Glue ofrece un parámetro para desactivar este comportamiento. Para obtener más información, consulte Parámetros de trabajo

utilizados por AWS Glue. AWS Glue seguirá utilizando el proxy local para publicar registros de CloudWatch de los trabajos de AWS Glue.

> **Note**
>
> - Esta característica es compatible con trabajos de AWS Glue en AWS Glue versión 2.0 y superior. Cuando utilice esta característica, debe asegurarse de que la VPC haya configurado una ruta a Simple Storage Service (Amazon S3) a través de un punto de conexión de VPC de servicio o NAT.
> - El parámetro de trabajo obsoleto `disable-proxy` solo enruta sus llamadas a Amazon S3 para descargar scripts y bibliotecas a través de su VPC. En cambio, se recomienda utilizar el nuevo parámetro `disable-proxy-v2`.

Ejemplo de uso

Crear un trabajo de AWS Glue con `disable-proxy-v2`:

```
aws glue create-job \
    --name no-proxy-job \
    --role GlueDefaultRole \
    --command "Name=glueetl,ScriptLocation=s3://my-bucket/glue-script.py" \
    --connections Connections="traffic-monitored-connection" \
    --default-arguments '{"--disable-proxy-v2" : "true"}'
```

Conexión a un almacén de datos de JDBC en una VPC

Normalmente, estos recursos se crean dentro de Amazon Virtual Private Cloud (Amazon VPC) de forma que no se pueda tener acceso a ellos a través de la red pública de Internet. De forma predeterminada, AWS Glue no puede tener acceso a los recursos dentro de una VPC. Para que AWS Glue pueda obtener acceso a los recursos dentro de su VPC, debe proporcionar información de configuración específica de VPC adicional que incluya los ID de subred y los ID de los grupos de seguridad de la VPC. AWS Glue utiliza esta información para configurar interfaces de red elásticas que permiten que la función se conecte de forma segura a otros recursos en la VPC privada.

Cuando utilice un punto de conexión de VPC, agréguelo a su tabla de enrutamiento. Para obtener más información, consulte Creación de un punto de conexión de VPC de interfaz para AWS Glue y Requisitos previos.

Cuando utilice el cifrado en el catálogo de datos, cree el punto de conexión de la interfaz KMS y agréguelo a la tabla de enrutamiento. Para obtener más información, consulte Crear un punto de conexión de VPC para AWS KMS.

Acceso a datos de VPC mediante interfaces de red elásticas

Al conectarse AWS Glue a un almacén de datos de JDBC en una VPC, AWS Glue crea una interfaz de red elástica (con el prefijo `Glue_`) en su cuenta para obtener acceso a sus datos de VPC. No puede eliminar esta interfaz de red siempre que se asocie a AWS Glue. Como parte de la creación de la interfaz de red elástica, AWS Glue asocia uno o varios grupos de seguridad a ella. Para permitir que AWS Glue cree la interfaz de red, los grupos de seguridad asociados al recurso deben permitir el acceso de entrada con una regla de origen. Esta regla contiene un grupo de seguridad que se asocia al recurso. Esto proporciona a la interfaz de red elástica acceso a su almacén de datos con el mismo grupo de seguridad.

Para que AWS Glue pueda comunicarse con sus componentes, especifique un grupo de seguridad con una regla de entrada con autorreferencia para todos los puertos TCP. Si crea una regla de entrada con autorreferencia, puede restringir el origen al mismo grupo de seguridad de la VPC y no abrirlo a todas las redes. Puede que el grupo de seguridad predeterminado de la VPC ya tenga una regla de entrada con autorreferencia para `ALL Traffic`.

Puede crear reglas en la consola de Amazon VPC. Para actualizar la configuración de la regla a través de la AWS Management Console, vaya a la consola de VPC (https://console.aws.amazon.com/vpc/) y seleccione el grupo de seguridad adecuado. Especifique la regla de entrada para que `ALL TCP` tenga como su origen el mismo nombre de grupo de seguridad. Para obtener más información acerca de las reglas de grupos de seguridad, consulte Grupos de seguridad de su VPC.

A cada interfaz de red elástica se le asigna una dirección IP privada del intervalo de direcciones IP de las subredes especificadas. No se asigna a la interfaz de red ninguna dirección IP pública. AWS Glue requiere acceso de Internet (por ejemplo, para obtener acceso a servicios de AWS que carecen de puntos de enlace de la VPC). Puede configurar una instancia de traducción de las direcciones de red (NAT) dentro de su VPC o puede usar la gateway NAT de Amazon VPC. Para obtener más información, consulte Gateways de NAT en la Guía del usuario de Amazon VPC. No puede usar directamente una gateway de Internet asociada a su VPC como ruta en su tabla de ruteo de la subred porque eso requiere que la interfaz de red tenga direcciones IP públicas.

Los atributos de red de VPC `enableDnsHostnames` y `enableDnsSupport` deben establecerse en true. Para obtener más información, consulte Utilización de DNS con su VPC.

> **Important**
>
> No ponga su almacén de datos en una subred pública o en una subred privada que no tenga acceso a Internet. En su lugar, asóciela únicamente a subredes privadas con acceso a Internet a través de una instancia NAT o una gateway NAT de Amazon VPC.

Propiedades de la interfaz de red elástica

Para crear la interfaz de red elástica, debe proporcionar las siguientes propiedades:

VPC

El nombre de la VPC que contiene su almacén de datos.

Subred

La subred de la VPC que contiene su almacén de datos.

Grupos de seguridad

Los grupos de seguridad que están asociados con el almacén de datos. AWS Glue asocia estos grupos de seguridad con la interfaz de red elástica que se asocia a su subred de VPC. Para permitir que los componentes de AWS Glue se comuniquen y también para evitar el acceso desde otras redes, al menos un grupo de seguridad elegido debe especificar una regla de entrada con autorreferencia para todos los puertos TCP.

Para obtener información acerca de cómo administrar una VPC con Amazon Redshift, consulte Administración de clústeres en una Amazon Virtual Private Cloud (VPC).

Para obtener más información sobre la administración de una VPC con Amazon Relational Database Service (Amazon RDS), consulte Trabajar con una instancia de base de datos de Amazon RDS en una VPC.

Uso de una conexión MongoDB o MongoDB Atlas

Después de crear una conexión para MongoDB o MongoDB Atlas puede utilizarla en un trabajo de ETL. Puede crear una tabla en la AWS Glue Data Catalog y especificar la conexión de MongoDB o MongoDB Atlas para el atributo `connection` de la tabla.

AWS Glue almacena la `url` y las credenciales de la conexión en la conexión de MongoDB. Los formatos de URI de conexión son los siguientes:

- Para MongoDB: mongodb://host:port/database. El host puede ser un nombre de host, una dirección IP o un socket de dominio UNIX. Si la cadena de conexión no especifica ningún puerto, utiliza el puerto predeterminado de MongoDB, 27017.
- Para MongoDB Atlas: mongodb+srv://server.example.com/database. El host puede ser un nombre de host que corresponde a un registro SRV de DNS. El formato SRV no requiere ningún puerto y utilizará el puerto MongoDB predeterminado, 27017.

Además, se pueden especificar opciones en el script del trabajo. Para obtener más información, consulte the section called “Conexión a MongoDB”.

Rastreo de un almacén de datos de Amazon S3 mediante un punto de conexión de VPC

Es posible que quiera configurar su almacén de datos de Amazon S3 o sus tablas del Catálogo de datos respaldadas por Amazon S3 para el acceso únicamente a través de un entorno de Amazon Virtual Private Cloud (Amazon VPC), con fines de seguridad, auditoría o control. En este tema se describe cómo crear y probar una conexión con el almacén de datos de Amazon S3 o las tablas del Catálogo de datos respaldadas por Amazon S3 en un punto de conexión de VPC mediante el tipo de conexión `Network`.

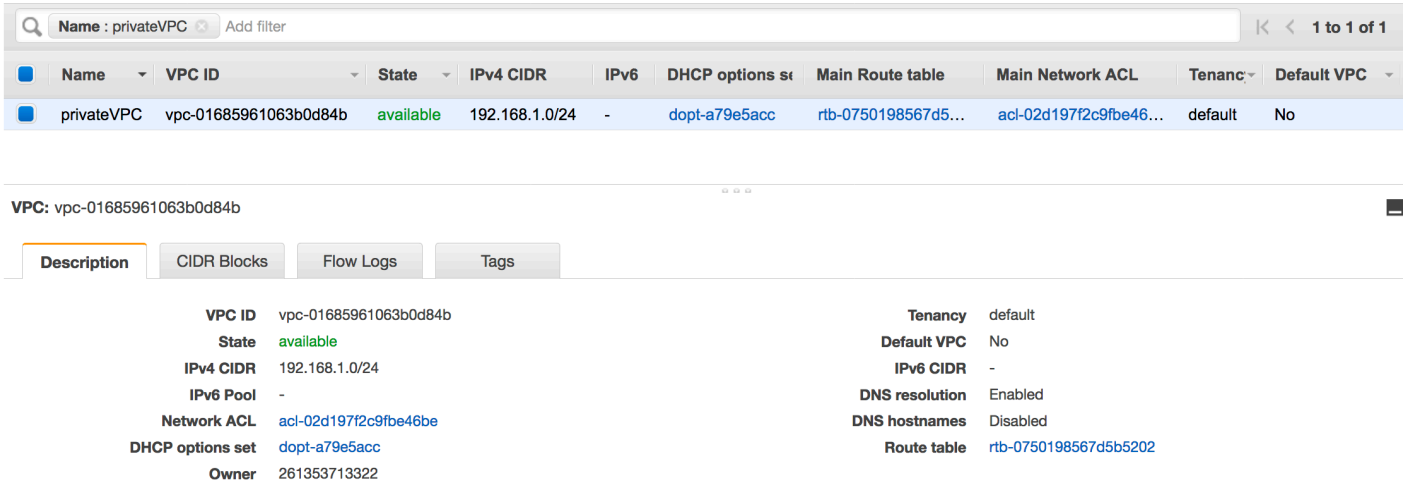
Realice las siguientes tareas para ejecutar un rastreador en el almacén de datos:

- the section called “Requisitos previos”
- the section called “Crear la conexión a Amazon S3”
- the section called “Prueba de la conexión a Amazon S3”
- the section called “Creación de un rastreador para un almacén de datos de Amazon S3”
- the section called “Ejecución de un rastreador”

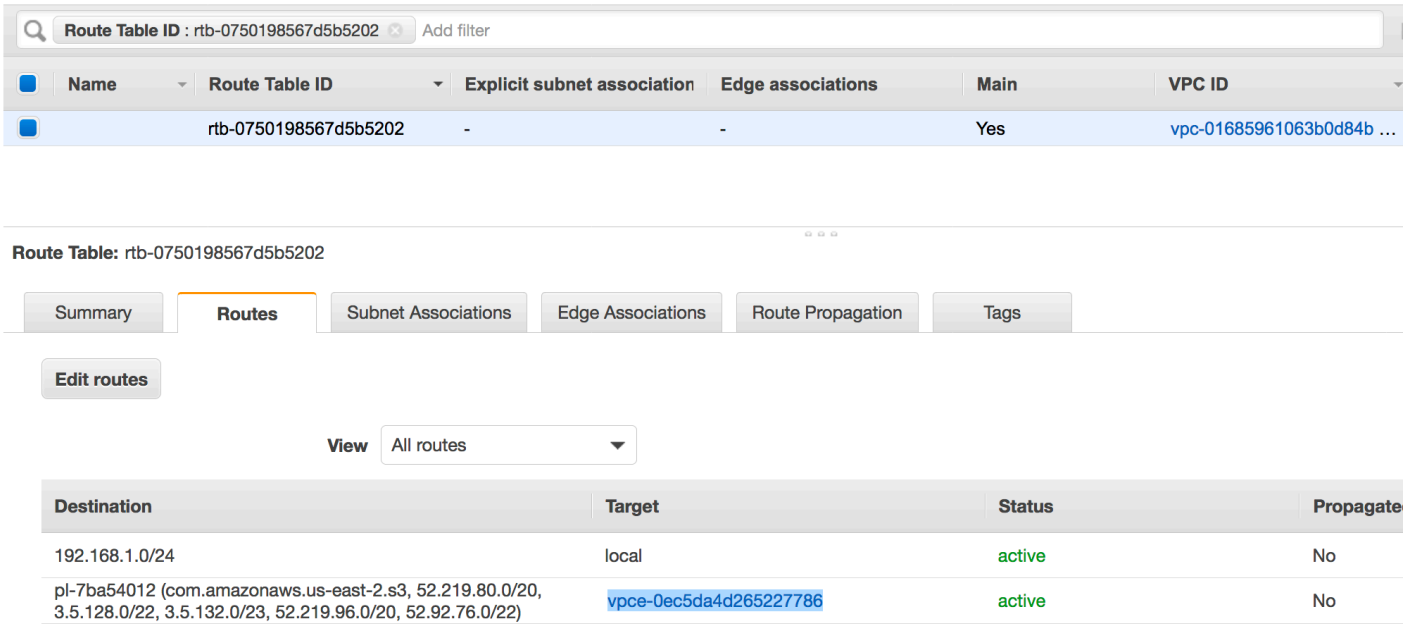
Requisitos previos

Verifique que ha cumplido estos requisitos previos para configurar el acceso a su almacén de datos de Amazon S3 o a sus tablas del Catálogo de datos respaldadas por Amazon S3 a través de un entorno de Amazon Virtual Private Cloud (Amazon VPC).

- Una VPC configurada. Por ejemplo: vpc-01685961063b0d84b. Para obtener más información, consulte Introducción a Amazon VPC en la Guía del usuario de Amazon VPC.
- Un punto de enlace de Amazon S3 asociado a la VPC. Por ejemplo: vpc-01685961063b0d84b. Para obtener más información, consulte Puntos de enlace para Amazon S3 en la Guía del usuario de Amazon VPC.



- Una entrada de ruta que apunta al punto de enlace de la VPC. Por ejemplo vpce-0ec5da4d265227786 en la tabla de enrutamiento utilizada por el punto de enlace de la VPC (vpce-0ec5da4d265227786).



- Una ACL de red asociada a la VPC permite el tráfico.
- Un grupo de seguridad asociado a la VPC permite el tráfico.

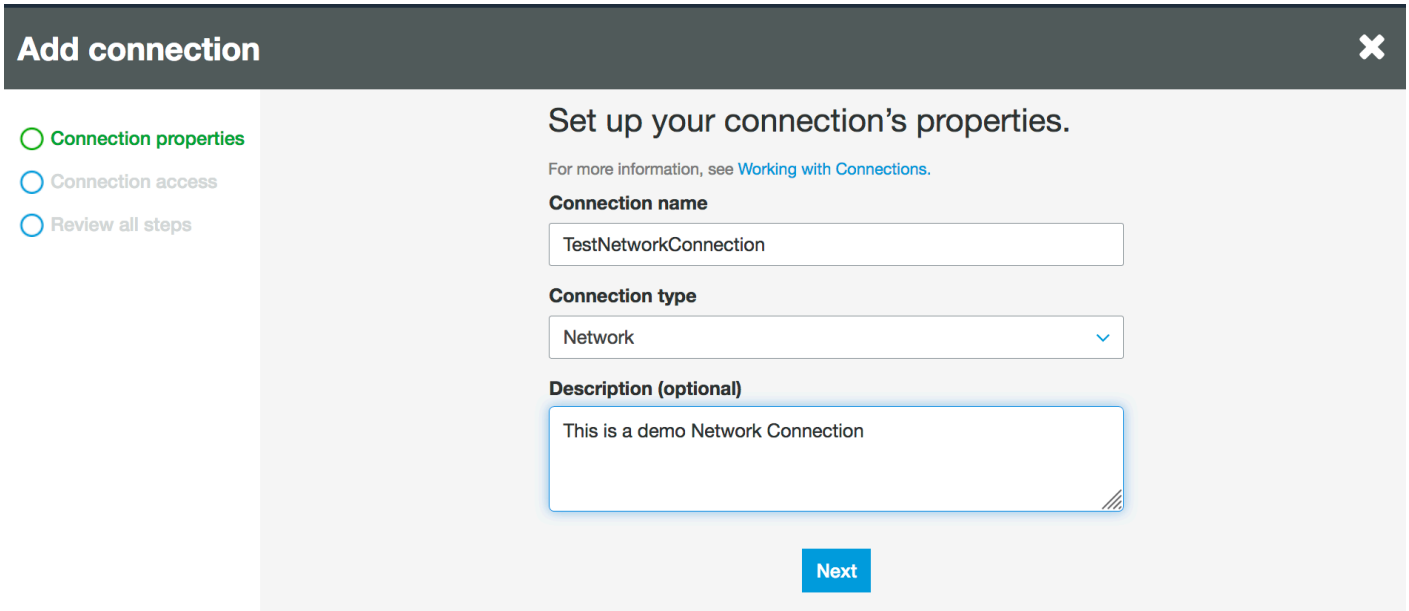
Crear la conexión a Amazon S3

Normalmente, estos recursos se crean dentro de Amazon Virtual Private Cloud (Amazon VPC) de forma que no se pueda tener acceso a ellos a través de la red pública de Internet. De forma predeterminada, AWS Glue no puede tener acceso a los recursos dentro de una VPC. Para que AWS Glue pueda obtener acceso a los recursos dentro de su VPC, debe proporcionar información de configuración específica de VPC adicional que incluya los ID de subred y los ID de los grupos de seguridad de la VPC. Para crear una conexión de `Network` deberá especificar la siguiente información:

- ID DE LA VPC
- Una subred dentro de la VPC
- Un grupo de seguridad

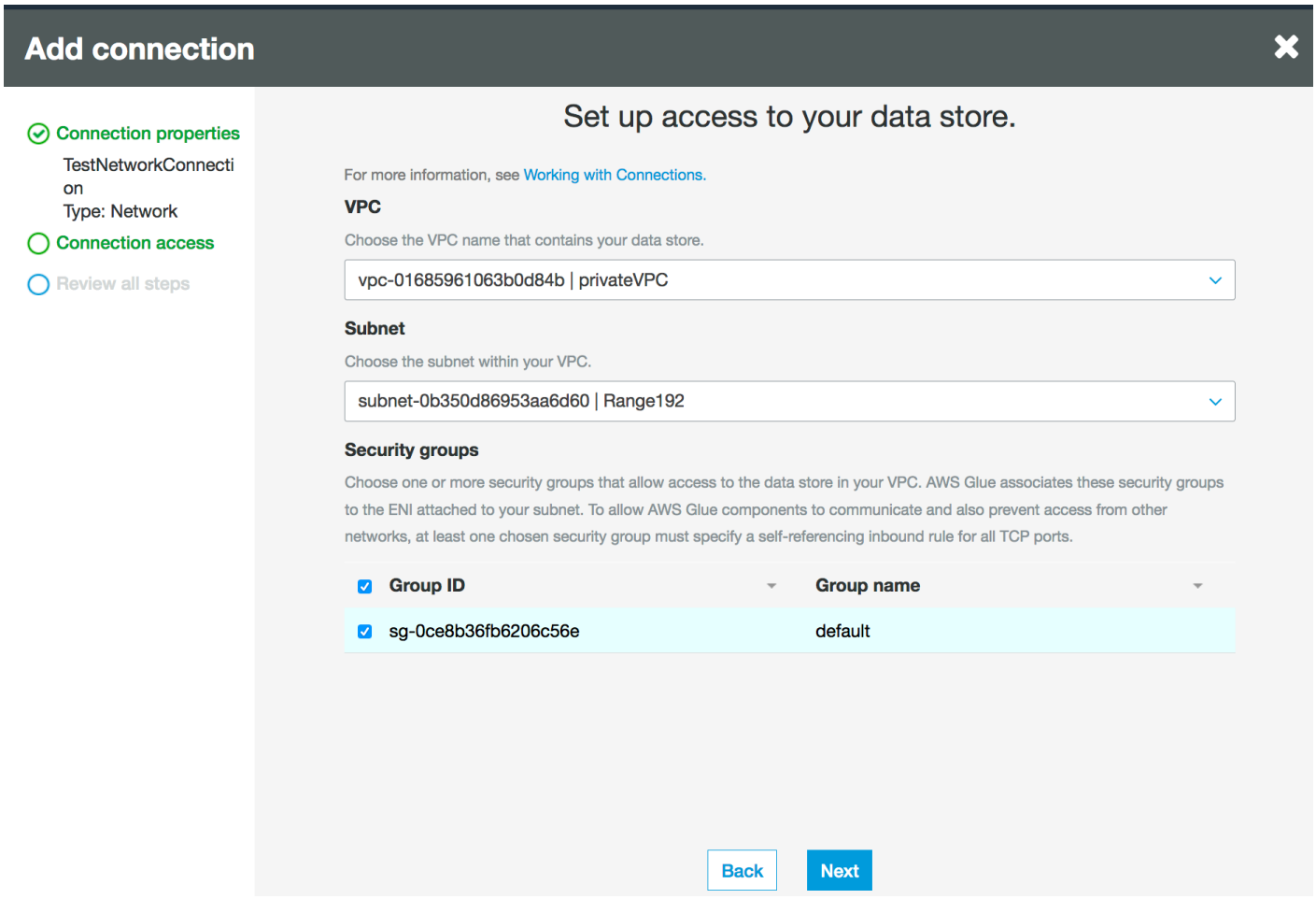
Para configurar una conexión `Network`:

1. Elija Add connection (Agregar conexión) en el panel de navegación de la consola de AWS Glue.
2. Ingrese el nombre de la conexión, elija Network (Red) como el tipo de conexión. Elija Next (Siguiente).

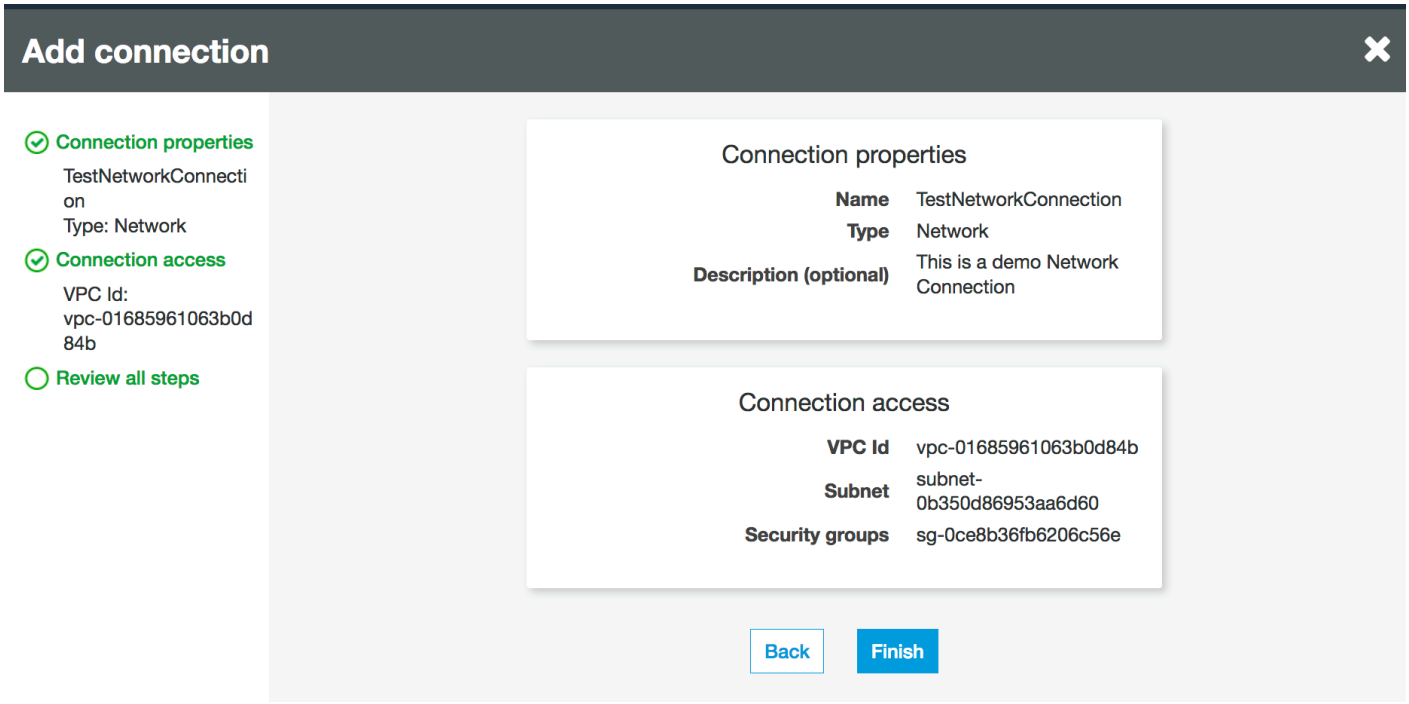


3. Configure la información de la VPC, subred y grupos de seguridad.
   - VPC: elija el nombre de la VPC que contiene su almacén de datos.
   - Subred: elija una subred en su VPC.

- Grupos de seguridad: elija uno o más grupos de seguridad que permitan el acceso al almacén de datos de la VPC.



4. Elija Next (Siguiente).
5. Verifique la información de conexión y elija Finish (Finalizar).



Prueba de la conexión a Amazon S3

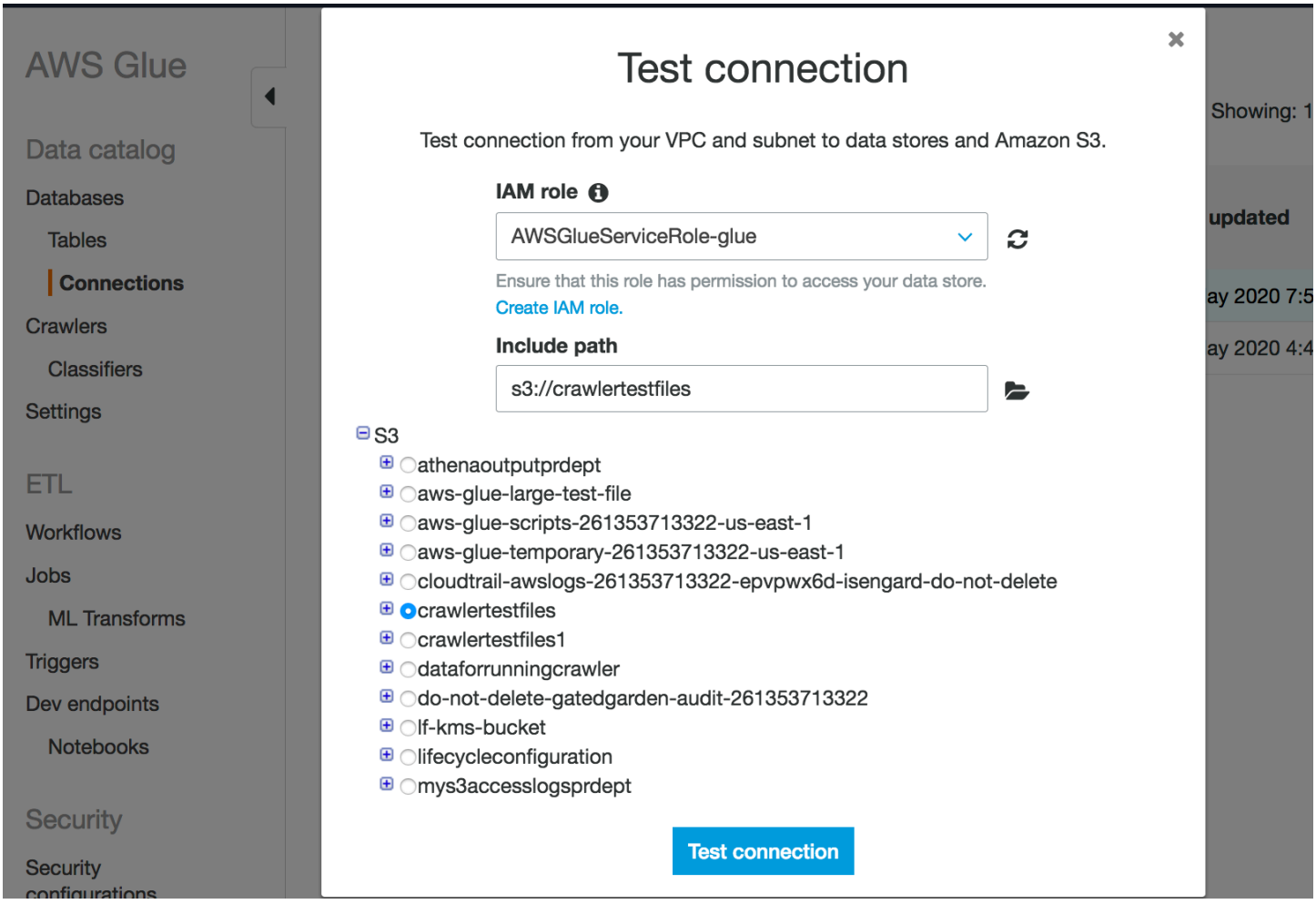
Una vez que haya creado su conexión `Network`, puede probar la conectividad con su almacén de datos de Amazon S3 en un punto de enlace de la VPC.

Pueden producirse los siguientes errores al probar una conexión:

- INTERNET CONNECTION ERROR (ERROR DE CONEXIÓN A INTERNET): indica un problema de conexión a Internet
- INVALID BUCKET ERROR (ERROR DE BUCKET NO VÁLIDO): indica un problema con el bucket de Amazon S3
- S3 CONNECTION ERROR (ERROR DE CONEXIÓN DE S3): indica un error al conectarse a Amazon S3
- INVALID CONNECTION TYPE (TIPO DE CONEXIÓN NO VÁLIDA): indica que el tipo de conexión no tiene el valor esperado, `NETWORK`
- INVALID CONNECTION TEST TYPE (TIPO DE PRUEBA DE CONEXIÓN NO VÁLIDA): indica un problema con el tipo de prueba de conexión de red
- INVALID TARGET (DESTINO NO VÁLIDO): indica que el bucket de Amazon S3 no se ha especificado correctamente

Para probar una conexión `Network`:

1. Seleccione la conexión Network (Red) en la consola de AWS Glue.
2. Elija Test Connection (Probar conexión).
3. Elija el rol de IAM que creó en el paso anterior y especifique un bucket de Amazon S3.
4. Elija Test connection (Probar conexión) para comenzar la prueba. Puede tardar unos minutos en mostrar el resultado.



Si recibe un error, verifique lo siguiente:

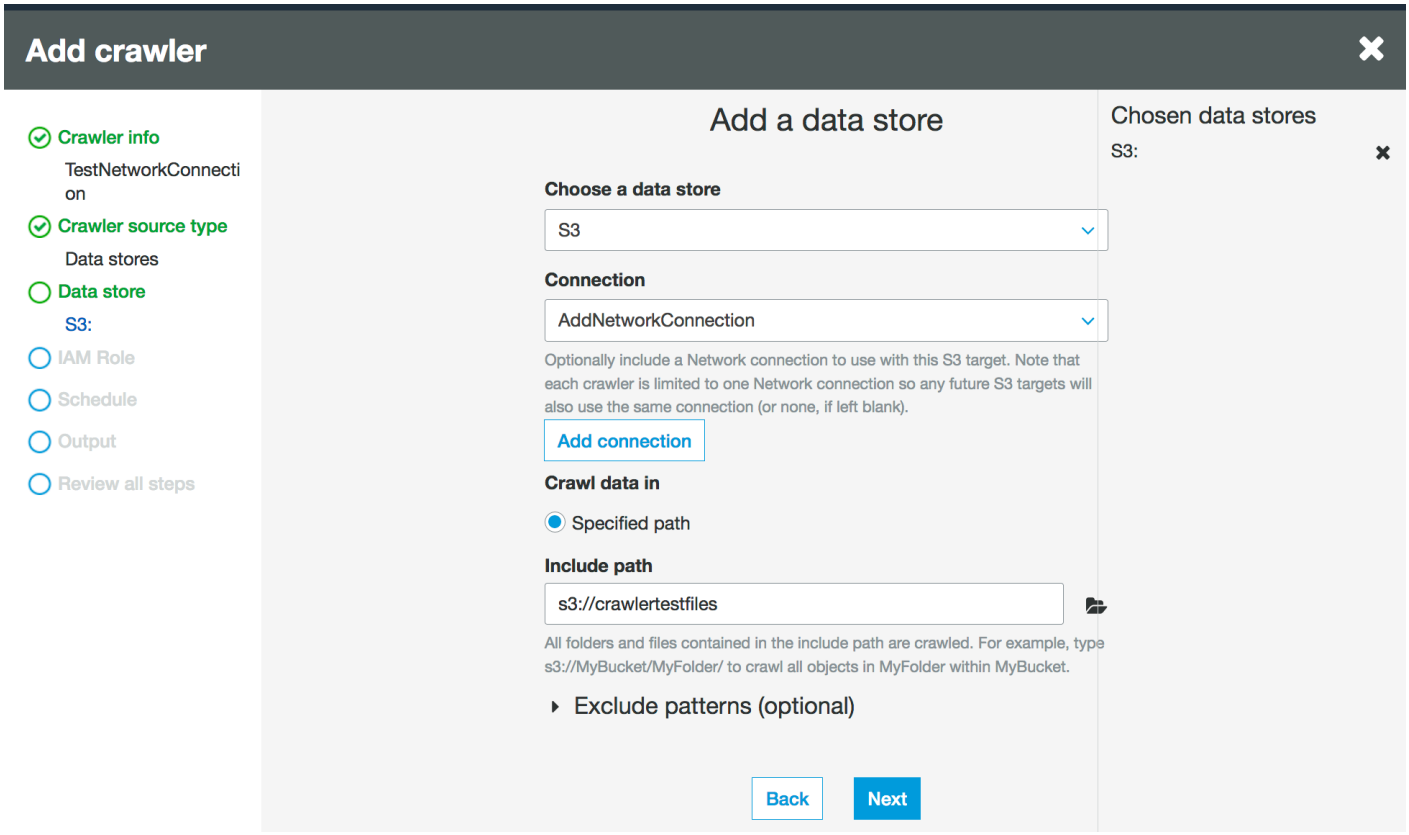
- Se proporcionan los privilegios correctos para el rol seleccionado.
- Se proporciona el bucket correcto de Amazon S3.
- Los grupos de seguridad y la ACL de red permiten el tráfico entrante y saliente requerido.
- La VPC especificada está conectada a un punto de conexión de VPC de Amazon S3.

Una vez haya probado la conexión con éxito, puede crear un rastreador.

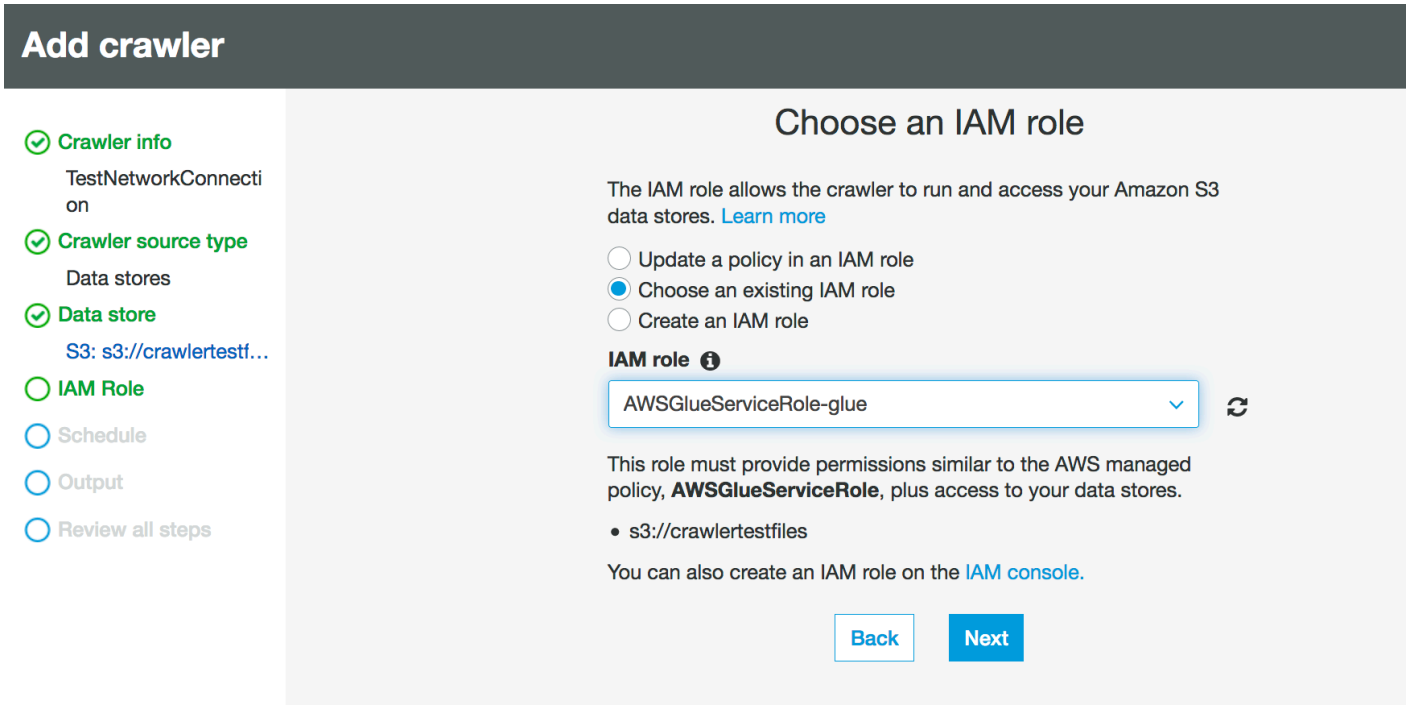
Creación de un rastreador para un almacén de datos de Amazon S3

Ahora, puede crear un rastreador que especifique la conexión `Network` que ha creado. Para obtener más información sobre cómo crear un rastreador, consulte Configuración de rastreadores.

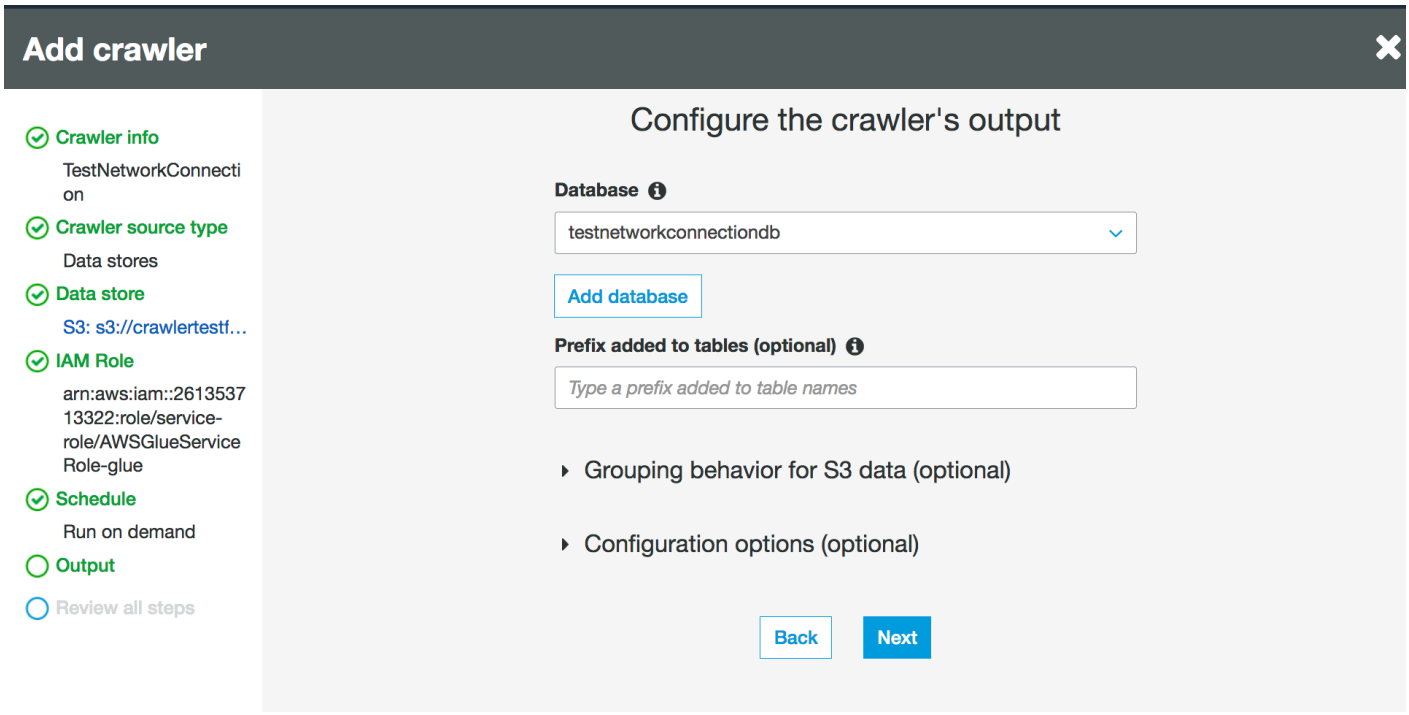
1. En primer lugar, elija Crawlers (Rastreadores) en el panel de navegación de la consola de AWS Glue.
2. Elija Add crawler (Agregar rastreador).
3. Especifique el nombre del rastreador y elija Next (Siguiente).
4. Cuando se le solicite el origen de los datos, elija S3 y especifique el prefijo del bucket de Amazon S3 y la conexión que creó con anterioridad.



5. Si es necesario, agregue otro almacén de datos en la misma conexión de red.
6. Elija el rol de IAM. El rol de IAM debe permitir el acceso al servicio de AWS Glue y al bucket de Amazon S3. Para obtener más información, consulte the section called “Configuración de rastreadores”.



7. Definir la programación para el rastreador.
8. Elija una base de datos existente en el Catálogo de datos o cree una nueva entrada de base de datos.

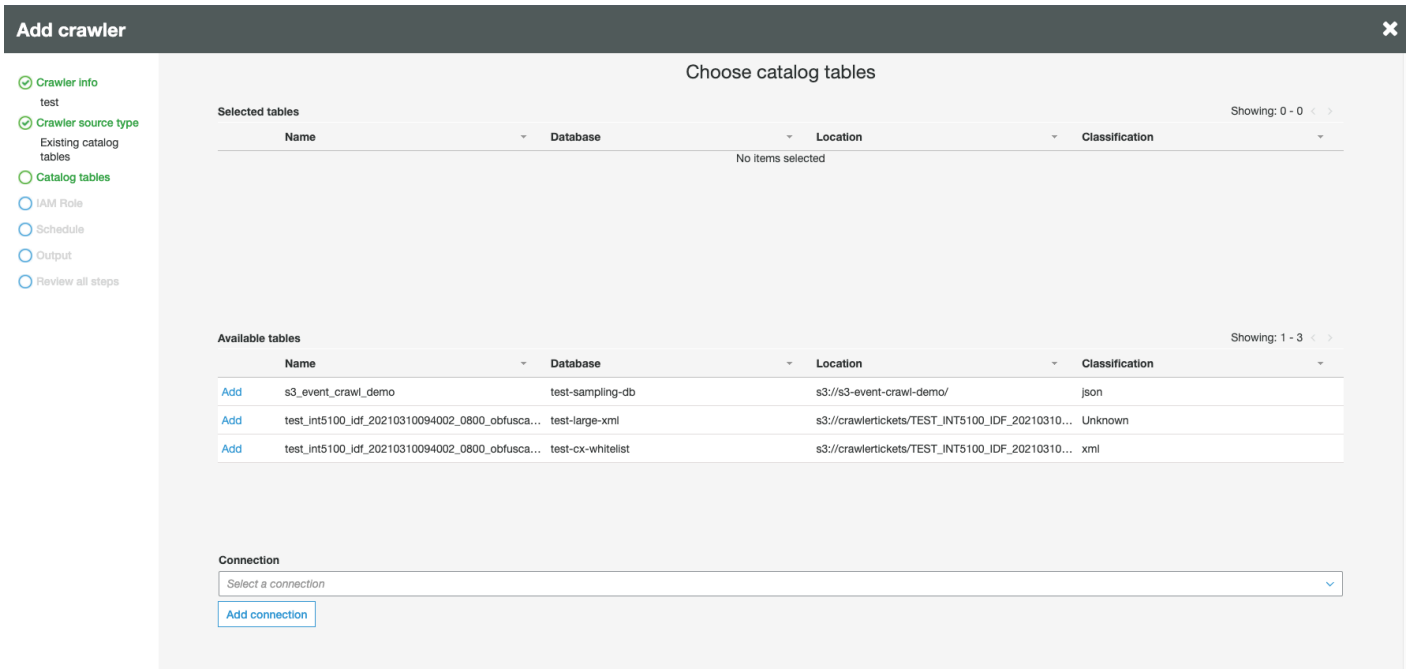


9. Finalice la configuración restante.

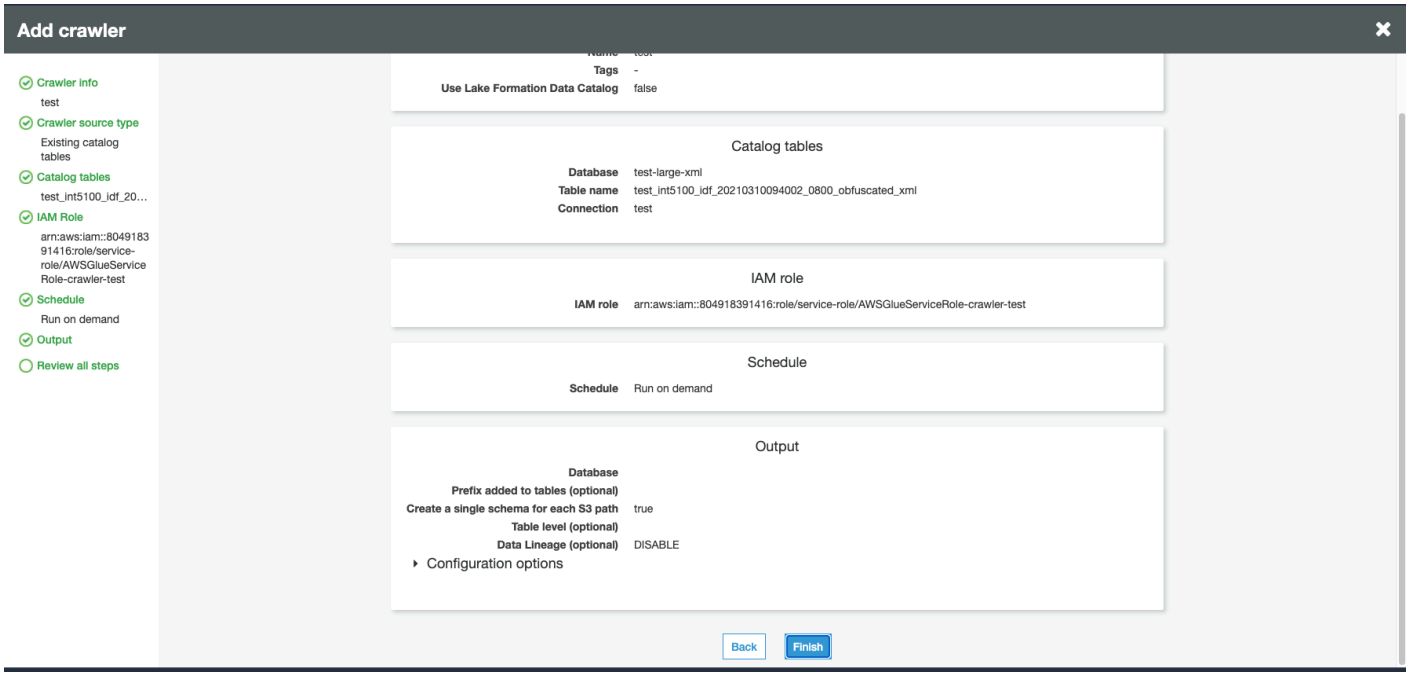
Creación de un rastreador para tablas del Catálogo de datos respaldadas por Amazon S3

Ahora, puede crear un rastreador que especifique la conexión `Network` que ha creado y un tipo de origen Catalog (Catálogo). Para obtener más información sobre cómo crear un rastreador, consulte Configuración de rastreadores.

1. En primer lugar, elija Crawlers (Rastreadores) en el panel de navegación de la consola de AWS Glue.
2. Elija Add crawler (Agregar rastreador).
3. Especifique el nombre del rastreador y elija Next (Siguiente).
4. Cuando se le solicite el tipo de origen del rastreador, elija Existing catalog tables (Tablas de catálogo existentes) y especifique las tablas de catálogo existentes que se van a rastrear en la lista de tablas disponibles.

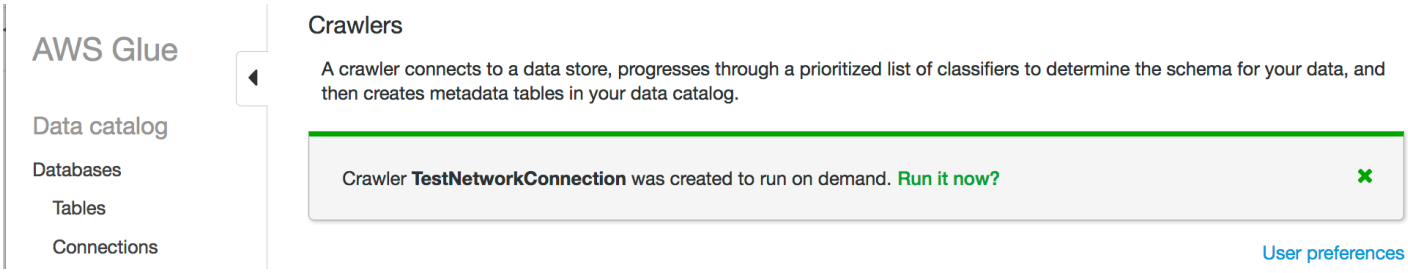


5. Elija el rol de IAM. El rol de IAM debe permitir el acceso al servicio de AWS Glue y al bucket de Amazon S3. Para obtener más información, consulte the section called “Configuración de rastreadores”.
6. Definir la programación para el rastreador.
7. Elija una base de datos existente en el Catálogo de datos o cree una nueva entrada de base de datos.
8. Finalice la configuración restante y revise los pasos.



Ejecución de un rastreador

Ejecute su rastreador.



Resolución de problemas

Para solucionar problemas relacionados con los buckets de Amazon S3 que utilizan una puerta de enlace de la VPC, consulte ¿Por qué no puedo conectarme a un bucket de S3 con un punto de enlace de la VPC de la gateway?

Solución de problemas de conexión en AWS Glue

Cuando un rastreador o un trabajo de AWS Glue utiliza las propiedades de conexión para obtener acceso a un almacén de datos, es posible que encuentre errores al intentar conectarse. AWS Glue utiliza direcciones IP privadas en la subred al crear interfaces de red elásticas especificadas en

su nube virtual privada (VPC) y subred. Los grupos de seguridad especificados en la conexión se aplican en cada una de las interfaces de red elásticas. Compruebe que los grupos de seguridad permiten el acceso saliente y si permiten la conectividad al clúster de base de datos.

Además, Apache Spark requiere conectividad bidireccional entre los nodos del controlador y el ejecutor. Uno de los grupos de seguridad debe permitir las reglas de entrada en todos los puertos TCP. Puede impedir que se abra al mundo restringiendo el origen del grupo de seguridad en sí mismo con un grupo de seguridad de autorreferencia.

A continuación le presentamos algunas acciones que puede emprender para solucionar problemas de conexión:

- Compruebe la dirección de puerto de su conexión.
- Compruebe el nombre de usuario y la cadena de contraseña de su conexión o secreto.
- Para un almacén de datos de JDBC, compruebe que permite conexiones entrantes.
- Compruebe que se puede obtener acceso a su almacén de datos dentro de su VPC.
- Si almacena las credenciales de la conexión mediante AWS Secrets Manager, asegúrese de que su rol de IAM para AWS Glue tenga permiso para acceder a su secreto. Para obtener más información, consulte Ejemplo: Permiso para recuperar valores de secretos en la Guía del usuario de AWS Secrets Manager. En función de la configuración de la red, es posible que también tenga que crear un punto de conexión de VPC para establecer una conexión privada entre la VPC y Secrets Manager. Para obtener más información, consulte Uso de un punto de conexión de VPC de AWS Secrets Manager.

Tutorial: uso de AWS Glue Connector for Elasticsearch

Elasticsearch es un motor de búsqueda y análisis popular y de código abierto para casos de uso como análisis de registros, monitoreo de aplicaciones en tiempo real y análisis de secuencias de clics. Puede usar OpenSearch como almacén de datos para sus trabajos de extracción, transformación y carga (ETL) si configura AWS Glue Connector for Elasticsearch en AWS Glue Studio. Este conector está disponible de forma gratuita en AWS Marketplace.

> **Note**
>
> AWS Marketplace Elasticsearch Spark Connector ha quedado obsoleto. En su lugar, use AWS Glue Connector for Elasticsearch.

En este tutorial, mostraremos cómo conectarse a sus nodos de Amazon OpenSearch Service con un número mínimo de pasos.

Temas

- Requisitos previos
- Paso 1: (opcional) cree un secreto de AWS para la información del clúster de OpenSearch
- Paso 2: suscríbase al conector
- Paso 3: activar el conector en AWS Glue Studio y crear una conexión
- Paso 4: configurar un rol de IAM para el trabajo de ETL
- Paso 5: crear un trabajo que utilice la conexión OpenSearch
- Paso 6: ejecutar el trabajo

Requisitos previos

Para utilizar este tutorial, debe disponer de lo siguiente:

- Acceso a AWS Glue Studio
- Acceso a un clúster de OpenSearch en AWS Cloud
- (Opcional) acceso a AWS Secrets Manager.

Paso 1: (opcional) cree un secreto de AWS para la información del clúster de OpenSearch

Para almacenar y utilizar de forma segura su credencial de conexión, guarde su credencial en AWS Secrets Manager. La conexión utilizará más tarde en el tutorial el secreto que haya creado. Los pares clave-valor de credenciales se introducirán en AWS Glue Connector for Elasticsearch como opciones de conexión normales.

Para obtener más información acerca de los permisos mínimos, consulte Creación y administración de secretos con AWS Secrets Manager en la Guía del usuario de AWS Secrets Manager.

Para crear un secreto de AWS

1. Inicie sesión en la consola de AWS Secrets Manager.
2. En la página de introducción del servicio o en la página con la lista Secretos, elija Store a new secret (Almacenar un nuevo secreto).

3. En la página Store a new secret (Almacenar un nuevo secreto), elija Other type of secret (Otro tipo de secreto). Esta opción indica que debe proporcionar la estructura y los detalles de su secreto.
4. Agregue una Clave y un Valor para el nombre de usuario del clúster de OpenSearch. Por ejemplo:

   `es.net.http.auth.user`: *`usuario`*
5. Seleccione + Add row (+ Agregar fila) e ingrese otro par clave-valor para la contraseña. Por ejemplo:

   `es.net.http.auth.pass`: *`contraseña`*
6. Elija Siguiente.
7. Escriba un nombre secreto. Por ejemplo: my-es-secret. Si lo desea, también puede escribir una descripción.

   Registre el nombre secreto, que se utilizará más adelante en este tutorial y, a continuación, elija Next (Siguiente).
8. Elija Next (Siguiente) una vez más, y luego elija Store (Almacenar) para crear el secreto.

Siguiente paso

Paso 2: suscríbase al conector

Paso 2: suscríbase al conector

AWS Glue Connector for Elasticsearch está disponible de manera gratuita en AWS Marketplace.

Para suscribirse a AWS Glue Connector for Elasticsearch en AWS Marketplace

1. Si aún no ha configurado su cuenta de AWS para utilizar License Manager, realice lo siguiente:

   a. Abra la consola de AWS License Manageren https://console.aws.amazon.com/license-manager.
   b. Elija Create customer managed license (Crear licencia administrada por el cliente).
   c. En la ventana IAM permissions (one-time setup) [Permisos de IAM (configuración única)], elija I grant AWS License Manager the required permissions (Concedo a Amazon License Manager los permisos necesarios), y, a continuación, elija Grant premissions (Conceder permisos).

Si no ve esta ventana, ya ha configurado los permisos necesarios.

2. Abra la consola de AWS Glue Studio en https://console.aws.amazon.com/gluestudio/.
3. En la consola de AWS Glue Studio, expanda el ícono del menú (☰) y luego elija Connectors (Conectores) en el panel de navegación.
4. En la página Connectors (Conectores), elija Go to AWS Marketplace (Ir a MKT).
5. En AWS Marketplace, en la sección Buscar productos de AWS Glue Studio, ingrese AWS Glue Connector for Elasticsearch en el campo de búsqueda y, luego, presione Intro.
6. Elija el nombre del conector, AWS Glue Connector for Elasticsearch.
7. En la página de producto del conector, utilice las pestañas para ver información sobre el conector. Cuando esté listo para continuar, elija Continue to Subscribe (Continuar con la suscripción).
8. Revise los términos de uso del software. Haga clic en Accept Terms (Aceptar términos).
9. Cuando finalice el proceso de suscripción, verá una notificación: “Thank you for subscribing to this product! You can now configure your software.” (Gracias por suscribirse a este producto. Ahora puede configurar su software). Arriba del banner encontrará el botón Continue to Configuration (Continuar con la configuración). Elija Continuar con la configuración.
10. Elija la opción Fulfillment (Ejecución) en la página Configure this software (Configurar este software). Puede elegir entre AWS Glue 1.0/2.0 o AWS Glue 3.0. A continuación, elija Continue to Launch (Continuar con el lanzamiento).

Siguiente paso

Paso 3: activar el conector en AWS Glue Studio y crear una conexión

Paso 3: activar el conector en AWS Glue Studio y crear una conexión

Después de elegir, Continue to Launch (Continuar con el lanzamiento), verá la página Launch this software (Lanzar este software) en AWS Marketplace. Se crea una conexión al utilizar el vínculo para habilitar el conector en AWS Glue Studio.

Para implementar el conector y crear una conexión en AWS Glue Studio

1. En la página Launch this software (Lanzar este software) en la consola de AWS Marketplace, elija Usage Instructions (Instrucciones de uso), y luego elija el enlace en la ventana que aparece.

Su navegador se redirige a la página Create marketplace connection (Crear conexión de marketplace) en la consola de AWS Glue Studio.

2. Escriba un nombre para la conexión. Por ejemplo: my-es-connection.
3. En la sección Connection access (Acceso a la conexión), para Connection credential type (Tipo de credenciales de conexión), elija User name and password (Nombre de usuario y contraseña).
4. En AWS secret (Secreto de AWS), especifique el nombre de su secreto. Por ejemplo: my-es-secret.
5. En la sección Network options (Opciones de red), ingrese la información de VPC para conectarse al clúster de OpenSearch.
6. Seleccione Create connection and activate connector (Crear conexión y habilitar conector).

Siguiente paso

Paso 4: configurar un rol de IAM para el trabajo de ETL

Paso 4: configurar un rol de IAM para el trabajo de ETL

Al crear el trabajo de ETL de AWS Glue, se especifica un rol de AWS Identity and Access Management (IAM) que el trabajo utilizará. El rol debe conceder acceso a todos los recursos utilizados por el trabajo, incluido Amazon S3 (para cualquier origen, destino, script, archivos de controladores y directorios temporales) y también objetos de AWS Glue Data Catalog.

El rol de IAM asumido para el trabajo de ETL de AWS Glue también debe tener acceso al secreto que se creó en la sección anterior. De forma predeterminada, el rol administrado de AWS, `AWSGlueServiceRole`, no tiene acceso al secreto. Para configurar el control de acceso para sus secretos, consulte Autenticación y control de acceso de AWS Secrets Manager y Limitación de acceso a secretos específicos.

Para configurar un rol de IAM para su trabajo de ETL

1. Configure los permisos descritos en the section called “Revisar los permisos de IAM necesarios para trabajos de ETL.”.
2. Configure los permisos adicionales necesarios al utilizar conectores con AWS Glue Studio, como se describe en the section called “Permisos necesarios para utilizar conectores”.

Siguiente paso

Paso 5: crear un trabajo que utilice la conexión OpenSearch

Paso 5: crear un trabajo que utilice la conexión OpenSearch

Después de crear un rol para su trabajo de ETL, puede crear un trabajo en AWS Glue Studio que utilice la conexión y el conector para Open Spark ElasticSearch.

Si su trabajo se ejecuta dentro de una Amazon Virtual Private Cloud (Amazon VPC), asegúrese de que la VPC esté configurada correctamente. Para obtener más información, consulte the section called “Configurar una VPC para su trabajo de ETL”.

Para crear un trabajo que utilice Elasticsearch Spark Connector

1. En AWS Glue Studio, elija Connectors (Conectores).
2. En la lista Your connections (Sus conexiones) seleccione la conexión que acaba de crear y elija Create job (Crear el trabajo).
3. En el editor visual de trabajos, elija el nodo Data source (Origen de datos). A la derecha, en la pestaña Data source properties - Connector (Propiedades del origen de datos: conector), configure información adicional para el conector.

    a. Seleccione Add schema (Agregar esquema) e ingrese el esquema del conjunto de datos en el origen de datos. Las conexiones no utilizan tablas almacenadas en Data Catalog, lo que significa que AWS Glue Studio no conoce el esquema de los datos. Debe proporcionar esta información del esquema en forma manual. Para obtener instrucciones sobre cómo utilizar el editor de esquemas, consulte the section called “Edición de esquema para un nodo de transformación personalizado”.

    b. Expanda Connection options (Opciones de conexión).

    c. Seleccione Add new option (Agregar nueva opción) e ingrese la información necesaria para el conector que no se ingresó en el secreto de AWS:

    - es.nodes: https://<OpenSearch domain endpoint>
    - es.port: 443
    - path: test
    - es.nodes.wan.only: true

Para obtener una explicación de estas opciones de conexión, consulte: https://www.elastic.co/guide/en/elasticsearch/hadoop/current/configuration.html.

4. Agregue un nodo de destino al gráfico.

   Su destino de datos puede ser Amazon S3 o puede usar información de AWS Glue Data Catalog o un conector para escribir datos en una ubicación diferente. Por ejemplo, puede utilizar una tabla del Data Catalog para escribir en una base de datos en Amazon RDS, o puede utilizar un conector como destino de datos para escribir en almacenes de datos que no son soportados de forma nativa en AWS Glue.

   Si elige un conector para el destino de datos, debe elegir una conexión creada para ese conector. Además, si el proveedor del conector lo requiere, debe agregar opciones para proporcionar información adicional al conector. Si utiliza una conexión que contiene información para un secreto de AWS, entonces no necesita proporcionar el nombre de usuario y la autenticación de contraseña en las opciones de conexión.
5. Si lo desea, agregue orígenes de datos adicionales y uno o más nodos de transformación como se describe en the section called “Edición de nodos de transformación de datos administrados por AWS Glue”.
6. Configure las propiedades del trabajo como se describe en the section called “Modificar las propiedades del trabajo”, comenzando con el paso 3, y guarde el trabajo.

Siguiente paso

Paso 6: ejecutar el trabajo

Paso 6: ejecutar el trabajo

Después de guardar el trabajo, puede ejecutar el trabajo para realizar las operaciones de ETL.

Para ejecutar el trabajo creado para AWS Glue Connector for Elasticsearch

1. Con la consola de AWS Glue Studio, en la página del editor visual, elija Run (Ejecutar).
2. En el banner de éxito, elija Run details (Detalles de la ejecución), o puede elegir la pestaña Runs (Ejecuciones) del editor visual para ver información sobre la ejecución del trabajo.

Creación de trabajos de AWS Glue con sesiones interactivas

Los ingenieros de datos pueden crear trabajos de AWS Glue de forma más rápida y sencilla que antes mediante el uso de las sesiones interactivas en AWS Glue.

Temas

- Información general sobre las sesiones interactivas de AWS Glue
- Introducción a las sesiones interactivas de AWS Glue
- Configuración de las sesiones interactivas de AWS Glue para cuadernos de Jupyter y AWS Glue Studio
- Introducción a las sesiones interactivas de AWS Glue For Ray (versión preliminar)
- Sesiones interactivas con IAM
- Conversión de un script o cuaderno en un trabajo de AWS Glue
- Sesiones interactivas de AWS Glue para streaming
- Desarrollo y prueba de scripts de trabajos de AWS Glue a nivel local
- Puntos de conexión de desarrollo

Información general sobre las sesiones interactivas de AWS Glue

Con las sesiones interactivas de AWS Glue, puede crear, probar y ejecutar rápidamente aplicaciones de análisis y preparación de datos. Las sesiones interactivas proporcionan una interfaz programática y visual para crear y probar scripts de extracción, transformación y carga (ETL) para la preparación de datos. Las sesiones interactivas ejecutan aplicaciones de análisis de Apache Spark y proporcionan acceso bajo demanda a un entorno en tiempo de ejecución remoto de Spark. AWS Glue administra de forma transparente Spark sin servidor para estas sesiones interactivas.

Dado que las sesiones interactivas son flexibles, puede crear y probar aplicaciones desde el entorno que elija. Puede crear sesiones interactivas y trabajar con ellas a través de AWS Command Line Interface y la API. Puede utilizar los cuadernos compatibles con Jupyter para crear y probar los scripts de su cuaderno en forma visual. Las sesiones interactivas proporcionan un kernel de Jupyter de código abierto que se integra prácticamente con cualquier entorno con el que Jupyter se integre, incluso con IDE como PyCharm, IntelliJ y VS Code. Esto le permite crear código en su entorno local y ejecutarlo sin problemas en el backend de sesiones interactivas.

Con la API de sesiones interactivas, los clientes pueden ejecutar mediante programación aplicaciones que utilizan análisis de Apache Spark sin la necesidad de administrar la infraestructura de Spark. Puede ejecutar una o varias instrucciones de Spark en una única sesión interactiva.

Por lo tanto, las sesiones interactivas proporcionan una forma más rápida, económica y flexible de crear y ejecutar aplicaciones de análisis y preparación de datos. Para aprender a utilizar las sesiones interactivas, consulte la documentación de esta sección. Comandos mágicos admitidos por AWS Glue

Limitaciones

- Los marcadores de trabajo no se admiten en las sesiones interactivas.
- No se admite la creación de trabajos de bloc de notas mediante AWS Command Line Interface.

Introducción a las sesiones interactivas de AWS Glue

En estas secciones, se describe cómo ejecutar sesiones interactivas de AWS Glue de forma local.

Requisitos previos para configurar las sesiones interactivas de manera local

A continuación, se indican los requisitos previos para instalar sesiones interactivas:

- Las versiones compatibles de Python son de la 3.6 a la 3.10+.
- Consulte las secciones siguientes para obtener instrucciones para macOS, Linux y Windows.

Instalación de Jupyter y sesiones AWS Glue interactivas (kernels de Jupyter)

Use lo siguiente para instalar el kernel localmente.

El comando, `install-glue-kernels`, instala la especificación de kernel de jupyter para los kernels pyspark y spark y también instala los logotipos en el directorio correcto.

```
pip3 install --upgrade jupyter boto3 aws-glue-sessions
```

```
install-glue-kernels
```

Ejecución de Jupyter

Para ejecutar el cuaderno de Jupyter, complete los siguientes pasos.

1. Para lanzar el cuaderno de Jupyter, ejecute el siguiente comando.

```
jupyter notebook
```

2. Elija New (Nuevo) y, a continuación, elija uno de los kernels de AWS Glue para comenzar a escribir código en AWS Glue.

Configuración de credenciales de sesión y región

Instrucciones para macOS/Linux

Las sesiones interactivas de AWS Glue requieren los mismos permisos de IAM que los trabajos y los puntos de conexión de desarrollo de AWS Glue. Especifique el rol que se utiliza con las sesiones interactivas de una de estas dos formas:

1. Con los comandos mágicos `%iam_role` y `%region`
2. Con una línea adicional en `~/.aws/config`

Configuración de un rol de sesión con un comando mágico

En la primera celda, escriba `%iam_role <YourGlueServiceRole>` en la primera celda que se ejecuta.

Configuración de un rol de sesión con **`~/.aws/config`**

AWS GlueEl rol de servicio para las sesiones interactivas puede especificarse en el propio cuaderno o guardarse junto con la configuración. AWS CLI Si tiene un rol que utiliza normalmente con los trabajos de AWS Glue, este será ese rol. Si no tiene un rol que utilice para los trabajos de AWS Glue, siga esta guía, Configuración de permisos de IAM para AWS Glue, para configurar uno.

Para establecer este rol como rol predeterminado de las sesiones interactivas:

1. Con un editor de texto, abra `~/.aws/config`.

2. Busque el perfil que utiliza para AWS Glue. Si no utiliza un perfil, use el perfil `[Default]`.
3. Agregue una línea en el perfil para el rol que quiera utilizar, como `glue_role_arn=<AWSGlueServiceRole>`.
4. [Opcional]: Si el perfil no tiene un conjunto de regiones predeterminadas, se recomienda agregar uno con `region=us-east-1` y reemplazar `us-east-1` con la región deseada.
5. Guarde la configuración.

Para obtener más información, consulte Sesiones interactivas con IAM.

Instrucciones para Windows

Las sesiones interactivas de AWS Glue requieren los mismos permisos de IAM que los trabajos y los puntos de conexión de desarrollo de AWS Glue. Especifique el rol que se utiliza con las sesiones interactivas de una de estas dos formas:

1. Con los comandos mágicos `%iam_role` y `%region`
2. Con una línea adicional en `~/.aws/config`

Configuración de un rol de sesión con un comando mágico

En la primera celda, escriba `%iam_role <YourGlueServiceRole>` en la primera celda que se ejecuta.

Configuración de un rol de sesión con **`~/.aws/config`**

AWS GlueEl rol de servicio para las sesiones interactivas puede especificarse en el propio bloc de notas o guardarse junto con la AWS CLI configuración. Si tiene un rol que utiliza normalmente con los trabajos de AWS Glue, este será ese rol. Si no tiene un rol que utiliza para los trabajos de AWS Glue, siga esta guía, Configuración de permisos de IAM para AWS Glue, para configurar uno.

Para establecer este rol como rol predeterminado de las sesiones interactivas:

1. Con un editor de texto, abra `~/.aws/config`.
2. Busque el perfil que utiliza para AWS Glue. Si no utiliza un perfil, use el perfil `[Default]`.
3. Agregue una línea en el perfil para el rol que quiera utilizar, como `glue_role_arn=<AWSGlueServiceRole>`.
4. [Opcional]: Si el perfil no tiene un conjunto de regiones predeterminadas, se recomienda agregar uno con `region=us-east-1` y reemplazar `us-east-1` con la región deseada.

5. Guarde la configuración.

Para obtener más información, consulte Sesiones interactivas con IAM.

Actualización desde la versión preliminar de las sesiones interactivas

El kernel se actualizó con nuevos nombres cuando se lanzó con la versión 0.27. Para limpiar las versiones preliminares de los núcleos, ejecute lo siguiente desde una terminal o PowerShell.

> **Note**
>
> Si forma parte de cualquier otra versión preliminar de AWS Glue que requiera un modelo de servicio personalizado, al eliminar el kernel se eliminará también el modelo de servicio personalizado.

```
# Remove Old Glue Kernels
jupyter kernelspec remove glue_python_kernel
jupyter kernelspec remove glue_scala_kernel

# Remove Custom Model
cd ~/.aws/models
rm -rf glue/
```

Uso de las sesiones interactivas con SageMaker Studio

Interactive Sessions de AWS Glue es un entorno de tiempo de ejecución bajo demanda y sin servidor de Apache Spark, el cual es utilizado por los científicos de datos para crear, probar y ejecutar de manera rápida la preparación de datos y las aplicaciones analíticas. Puede iniciar una sesión interactiva de AWS Glue al iniciar un cuaderno de Amazon SageMaker Studio Classic.

Para obtener más información, consulte Preparación de datos con las sesiones interactivas de AWS Glue.

Uso de las sesiones interactivas con Microsoft Visual Studio Code

Requisitos previos

- Instale las sesiones interactivas de AWS Glue y verifique que funcionan con Jupyter Notebook.
- Descargue e instale Visual Studio Code con Jupyter. Para obtener detalles, consulte Jupyter Notebook en VS Code

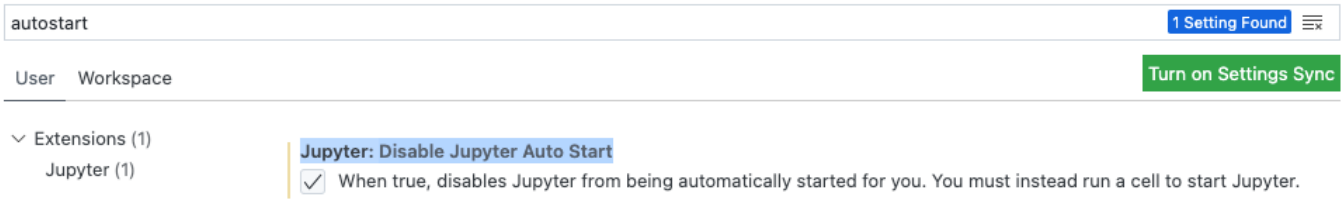
Para empezar con las sesiones interactivas con VSCode

1. Desactive Jupyter AutoStart en VS Code.

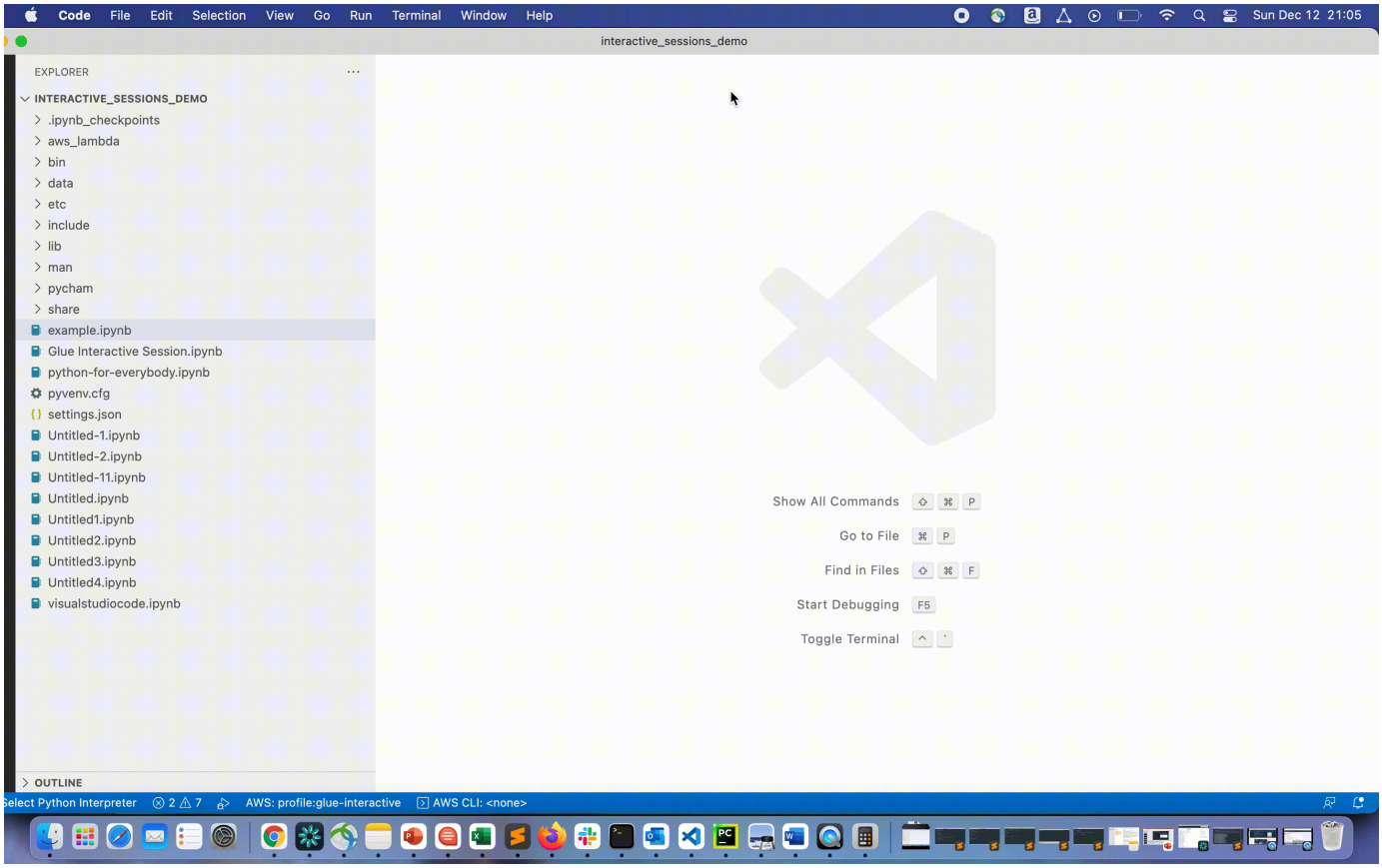
   En Visual Studio Code, los kernels de Jupyter se iniciarán automáticamente, lo que evitará que los comandos mágicos surtan efecto, ya que la sesión ya se habrá iniciado. Para deshabilitar el inicio automático en Windows, vaya a Archivo > Preferencias > Extensiones > Jupyter > haga clic con el botón derecho en Jupyter y, a continuación, seleccione Configuración de extensión.

   En MacOS, vaya a Código > Configuración > Extensiones > Jupyter >, haga clic con el botón derecho en Jupyter y, a continuación, seleccione Configuración de extensión.

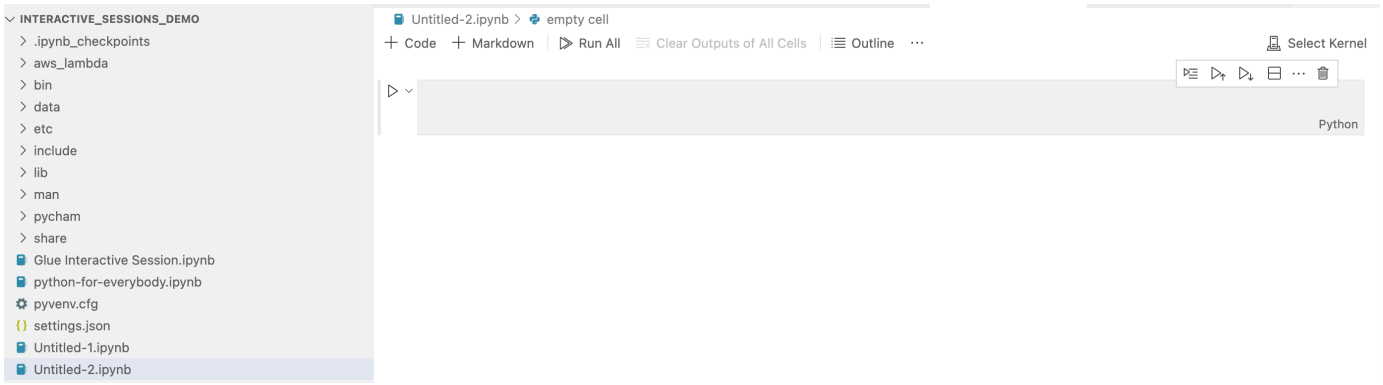
   Desplácese hacia abajo hasta que vea Jupyter: deshabilitar el inicio automático de Jupyter. Marque la casilla “Si es verdadero, se deshabilita el inicio automático de Jupyter”. You must instead run a cell to start Jupyter” (Si es verdadero, desactiva el inicio automático de Jupyter. En su lugar, se debe ejecutar una celda para iniciar Jupyter).



2. Vaya a File (Archivo) > New File (Nuevo archivo) > Save (Guardar) para guardar este archivo con el nombre que elija como extensión de `.ipynb`, o bien seleccione jupyter en Select a language (Seleccionar un idioma) y guarde el archivo.



3. Haga doble clic en el archivo. Aparecerá el shell de Jupyter y se abrirá un cuaderno.



4. En Windows, cuando se crea un archivo por primera vez, de manera predeterminada no tiene seleccionado ningún kernel. Haga clic en Select Kernel (Seleccionar kernel) y se mostrará una lista de los kernels disponibles. Elija Glue PySpark.

   En MacOS, si no ve el kernel Glue PySpark, siga estos pasos:

   1. Ejecute una sesión local de Jupyter para obtener la URL.

      Por ejemplo, para lanzar el cuaderno de Jupyter, ejecute el siguiente comando.

```
jupyter notebook
```

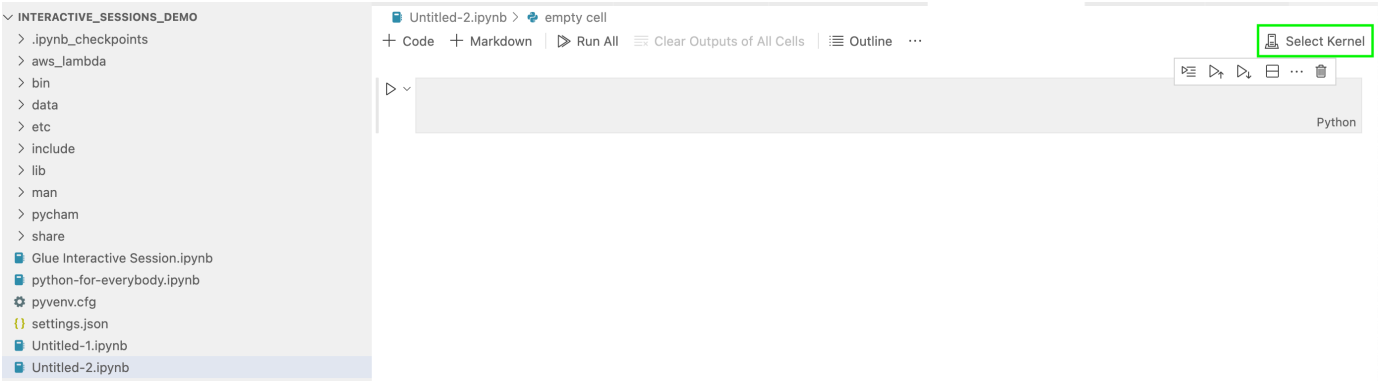
Cuando el cuaderno se ejecute por primera vez, verá una URL similar a `http://localhost:8888/?token=3398XXXXXXXXXXXXXXX`.

Copie la dirección URL.

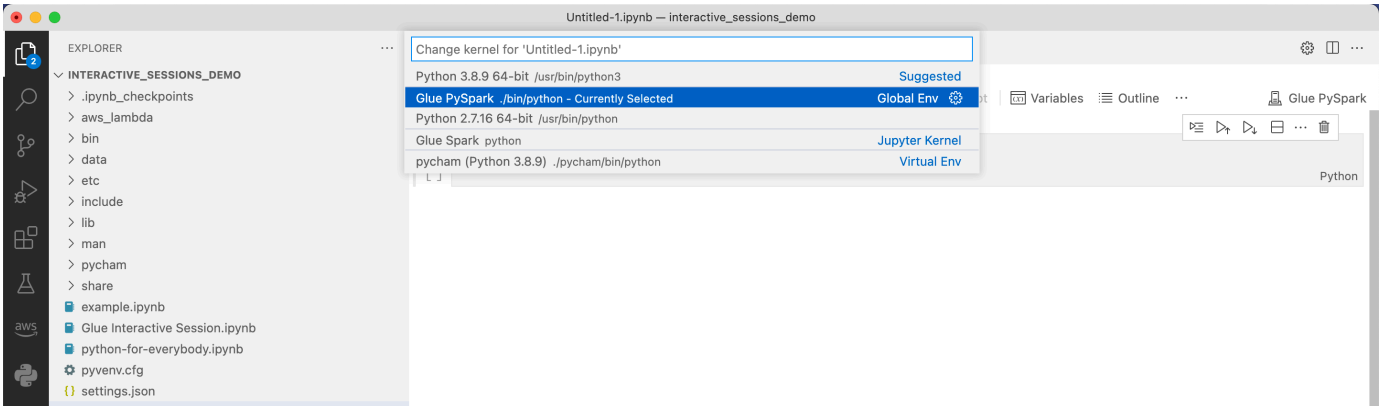
2. En VS Code, haga clic en el kernel actual, luego Seleccionar otro kernel... y, a continuación, seleccione Servidor de Jupyter existente.... Pegue la URL que copió en el paso anterior.

   Si recibe un mensaje de error, consulte la wiki de VS Code Jupyter.

3. Si tiene éxito, se configurará el kernel como Glue PySpark.



Elija el kernel Glue PySpark o Glue Spark(para Python y Scala respectivamente).



Si no aparecen los kerneles AWS Glue PySpark y AWS Glue Spark en la lista desplegable, asegúrese de que haya instalado el kernel de AWS Glue en el paso anterior, o de que la configuración `python.defaultInterpreterPath` en Visual Studio Code sea

correcta. Para obtener más información, consulte la Descripción de la configuración de python.defaultInterpreterPath.

5. Creación de uns sesión interactiva de AWS Glue. Proceda a crear una sesión del mismo modo en que lo hizo en el cuaderno de Jupyter. Especifique cualquier comando mágico en la parte superior de la primera celda y ejecute una instrucción de código.

Configuración de las sesiones interactivas de AWS Glue para cuadernos de Jupyter y AWS Glue Studio

Introducción a los comandos mágicos de Jupyter

Los comandos mágicos de Jupyter son comandos que se pueden ejecutar al principio de una celda o como un cuerpo de celda completo. Los comandos mágicos comienzan por `%` en el caso de los comandos mágicos de línea, y por `%%` si son comandos mágicos de celda. Los comandos mágicos de línea como `%region` y `%connections` se pueden ejecutar con varios comandos mágicos en una celda, o bien con código incluido en el cuerpo de la celda, como en el siguiente ejemplo.

```
%region us-east-2
%connections my_rds_connection
dy_f = glue_context.create_dynamic_frame.from_catalog(database='rds_tables',
 table_name='sales_table')
```

Los comandos mágicos de celda deben utilizar toda la celda y pueden hacer que el comando abarque varias líneas. A continuación aparece un ejemplo de `%%sql`.

```
%%sql
select * from rds_tables.sales_table
```

Comandos mágicos compatibles con las sesiones interactivas de AWS Glue para Jupyter

Los siguientes son comandos mágicos que se pueden utilizar con sesiones interactivas de AWS Glue para cuadernos de Jupyter.

Comandos mágicos de las sesiones

Nombre	Tipo	Descripción
`%help`	n/a	Devuelve una lista de descripciones y tipos de entrada para todos los comandos mágicos.
`%profile`	Cadena	Especifique un perfil en la AWS configura ción para usarlo como proveedor de credenciales.
`%region`	Cadena	Especifique el Región de AWS; en el que se va a inicializar una sesión. Valor predeterminado de `~/.aws/co nfigure`. Ejemplo: `%region us-west-1`
`%idle_timeout`	Int	Número de minutos de inactividad tras los que se agotará el tiempo de espera de una sesión una vez que se haya ejecutado una celda. El valor predeterminado de tiempo de inactividad para las sesiones de ETL de Spark es el tiempo de espera predeterm inado: 2880 minutos (48 horas). Para otros tipos de sesiones, consulte la documenta ción correspondiente. Ejemplo: `%idle_timeout 3000`
`%session_id`	n/a	Devuelve el ID de la sesión en ejecución.
`%session_id_prefix`	Cadena	Defina una cadena que precederá a todos los ID de sesión con el formato [session_ id_prefix]-[session_id] ([prefijo_id_sesión]-[id_sesión]). Si no se proporciona un ID de sesión, se generará un UUID aleatorio . Este comando mágico no se admite

Nombre	Tipo	Descripción
		cuando se ejecuta un cuaderno de Jupyter en AWS Glue Studio. Ejemplo: `%session_id_prefix 001`
`%status`		Devuelve el estado de la sesión actual de AWS Glue, incluyendo su duración, configuración y rol o usuario ejecutante.
`%stop_session`		Detiene la sesión actual.
`%list_sessions`		Se enumeran todas las sesiones que se están ejecutando actualmente por nombre e ID.
`%session_type`	Cadena	Establece el tipo de sesión en Streaming, ETL o Ray. Ejemplo: `%session_type Streaming`
`%glue_version`	Cadena	Versión de AWS Glue que debe utilizar esta sesión. Ejemplo: `%glue_version 3.0`

Comandos mágicos para seleccionar los tipos de trabajo

Nombre	Tipo	Descripción
`%streaming`	Cadena	Cambia el tipo de sesión a streaming de AWS Glue.
`%etl`	Cadena	Cambia el tipo de sesión a ETL de AWS Glue.

Nombre	Tipo	Descripción
%glue_ray	Cadena	Cambia el tipo de sesión a AWS Glue para Ray. Consulte las sesiones interactivas de Magics compatibles con AWS Glue Ray.

Comandos mágicos de configuración de AWS Glue para Spark

El comando mágico de `%%configure` es un diccionario en formato JSON que consta de todos los parámetros de configuración para una sesión. Cada parámetro se puede especificar aquí o mediante comandos mágicos individuales.

Nombre	Tipo	Descripción
`%%configure`	Diccionario	Especifica un diccionario en formato JSON que consta de todos los parámetros de configuración para una sesión. Cada parámetro se puede especificar aquí o mediante comandos mágicos individuales. Para ver una lista de parámetros y ejemplos de cómo usar `%%configure` , consulte la siguiente tabla: Uso de %%configure.
`%iam_role`	Cadena	Especifica un ARN de rol de IAM con el que ejecutar la sesión. Valor predeterminado de ~/.aws/configure. Ejemplo: `%iam_role AWSGlueServiceRole`
`%number_of_workers`	Int	Número de procesos de empleados de un worker_type definido que se asignan cuando se ejecuta un trabajo. También se debe configurar `worker_type` . El valor predeterminado de `number_of_workers` es 5.

Nombre	Tipo	Descripción
		Ejemplo: `%number_of_workers 2`
`%additional_python_modules`	Enumeración	Lista separada por comas de módulos de Python adicionales que se deben incluir en el clúster (pueden ser de PyPI o S3). Ejemplo:`%additional_python_modules pandas, numpy` .

Nombre	Tipo	Descripción

`%%tags`	Cadena	Agrega etiquetas a una sesión. Especifiq ue las etiquetas entre corchetes { }. Cada par de nombres de etiquetas está entre paréntesis (“”) y separado por una coma (,).
		
		`%%tags`
		`{"billing":"Data-Platform",`
		`"team":"analytics"}`
		
		Use el comando mágico `%status` para ver las etiquetas asociadas a la sesión.
		
		`%status`
		
		`Session ID: <sessionId>`
		`Status: READY`
		`Role: <example-role>`
		`CreatedOn: 2023-05-26 11:12:17.`
		`056000-07:00`
		`GlueVersion: 3.0`
		`Job Type: glueetl`
		`Tags: {'owner':'example-owner',`
		`'team':'analytics', 'billing'`
		`:'Data-Platform'}`
		`Worker Type: G.4X`
		`Number of Workers: 5`
		`Region: us-west-2`
		`Applying the following default`
		`arguments:`
		`--glue_kernel_version 0.38.0`
		`--enable-glue-datacatalog true`
		`Arguments Passed: ['--glue_`
		`kernel_version: 0.38.0', '--`
		`enable-glue-datacatalog: true']`

Nombre	Tipo	Descripción

`%%assume_role`	Diccionario	Especifique un diccionario en formato JSON o una cadena ARN de rol de IAM para crear una sesión para el acceso entre cuentas.
		
		Ejemplo con ARN:
		
		`%%assume_role`
		`{`
		`  'arn:aws:iam::XXXXXXXXXXXX:`
		`role/AWSGlueServiceRole'`
		`}`
		
		Ejemplo con credenciales:
		
		` %%assume_role`
		`{{`
		`    "aws_access_key_id" =`
		` "XXXXXXXXXXXX",`
		`    "aws_secret_access_key" =`
		` "XXXXXXXXXXXX",`
		`    "aws_session_token" =`
		` "XXXXXXXXXXXX"`
		`}}`

Argumentos de comandos mágicos celulares %%configure

El comando mágico de `%%configure` es un diccionario en formato JSON que consta de todos los parámetros de configuración para una sesión. Cada parámetro se puede especificar aquí o mediante comandos mágicos individuales. A continuación se muestran ejemplos de argumentos compatibles con el comando mágico celular `%%configure`. Utilice el prefijo `--` para los argumentos de ejecución especificados para el trabajo. Ejemplo:

```
%%configure
```

```
{
   "--user-jars-first": "true",
   "--enable-glue-datacatalog": "false"
}
```

Para obtener más información sobre los parámetros del trabajo, consulte Parámetros del flujo de trabajo.

Configuración de la sesión

Parámetro	Tipo	Descripción

`max_retries`	Int	Número máximo de reintentos permitidos para esta tarea si se genera un error.
		`%%configure`
		`{`
		`  "max_retries": "0"`
		`}`

`max_concurrent_runs`	Int	El número máximo de ejecuciones simultáneas que están permitidas para un trabajo.
		Ejemplo:
		`%%configure`
		`{`
		`  "max_concurrent_runs": "3"`
		`}`

Parámetros de la sesión

Parámetro	Tipo	Descripción

`--enable-spark-ui`	Booleano	Activa la interfaz de usuario de Spark para supervisar y depurar los trabajos de AWS Glue ETL.
		`%%configure`
		`{`
		`  "--enable-spark-ui": "true"`
		`}`

`--spark-event-logs-path`	Cadena	Especifica una ruta de Amazon S3. Al usar la característica de monitoreo de la interfaz de usuario de Spark.
		Ejemplo:
		`%%configure`
		`{`
		`  "--spark-event-logs-path":`
		`"s3://path/to/event/logs/"`
		`}`

`--script_location`	Cadena	Especifica la ruta de S3 a un script que ejecuta un flujo de trabajo.
		Ejemplo:
		`%%configure`
		`{`
		`  "script_location": "s3://new-folder-here"`
		`}`

`--SECURITY_CONFIGURATION`	Cadena	El nombre de una configuración de AWS Glue seguridad

Parámetro	Tipo	Descripción

		Ejemplo:
		`%%configure`
		`{`
		`"--SECURITY_CONFIG URATION": security-configuration-name ,`
		`}`

`--job-language`	Cadena	El lenguaje de programación del script. Acepta un valor de “scala” o “python”. El valor predeterminado es “python”.
		Ejemplo:
		`%%configure`
		`{`
		`"--job-language": "scala"`
		`}`

`--class`	Cadena	La clase de Scala que sirve de punto de entrada del script de Scala. El valor predeterminado es nulo.
		Ejemplo:
		`%%configure`
		`{`
		`"--class": "className"`
		`}`

Parámetro	Tipo	Descripción

`--user-jars-first`	Booleano	Prioriza los archivos JAR adicionales del cliente en la ruta de clases. El valor predeterminado es nulo.
		
		Ejemplo:
		
		`%%configure`
		`{`
		`  "--user-jars-first": "true"`
		`}`

`--use-postgres-driver`	Booleano	Prioriza el controlador JDBC de Postgres en la ruta de clase para evitar conflictos con el controlador JDBC. Amazon Redshift El valor predeterminado es nulo.
		
		Ejemplo:
		
		`%%configure`
		`{`
		`  "--use-postgres-driver": "true"`
		`}`

Parámetro	Tipo	Descripción

`--extra-files`	List(cadena)	Las rutas de Amazon S3 a archivos adicionales, por ejemplo, archivos de configuración que AWS Glue copia en el directorio de trabajo del script antes de ejecutarlo.
		
		Ejemplo:
		
		`%%configure`
		`{`
		`  "--extra-files": "s3://path/to/additional/files/"`
		`}`

`--job-bookmark-option`	Cadena	Controla el comportamiento de un marcador de trabajo. Acepta los valores '', 'o' '. job-bookmark-enable job-bookmark-disable job-bookmark-pause El valor predeterminado es 'job-bookmark-disable'.
		
		Ejemplo:
		
		`%%configure`
		`{`
		`  "--job-bookmark-option": "job-bookmark-enable"`
		`}`

Parámetro	Tipo	Descripción

`--TempDir`	Cadena	Especifica una ruta de Amazon S3 a un bucket que se puede utilizar como directorio temporal para el trabajo. El valor predeterminado es nulo.
		
		Ejemplo:
		
		`%%configure`
		`{`
		`  "--TempDir": "s3://path/to/temp/dir"`
		`}`

`--enable-s3-parquet-optimized-committer`	Booleano	Habilita el confirmador optimizado para Amazon S3 de EMRFS de forma que puedan escribirse datos de Parquet en Amazon S3. El valor predeterminado es “verdadero”.
		
		Ejemplo:
		
		`%%configure`
		`{`
		`  "--enable-s3-parquet-optimized-committer": "false"`
		`}`

Parámetro	Tipo	Descripción

`--enable-rename-algorithm-v2`	Booleano	Establece la versión del algoritmo de cambio de nombre de EMRFS a la versión 2. El valor predeterminado es “verdadero”.
		
		Ejemplo:
		
		`%%configure`
		`{`
		`  "--enable-rename-algorithm-v2": "true"`
		`}`

`--enable-glue-datacatalog`	Booleano	Permite utilizar el Catálogo de datos de AWS Glue como metaalmacén de Apache Spark Hive.
		
		Ejemplo:
		
		`%%configure`
		`{`
		`  --"enable-glue-datacatalog": "true"`
		`}`

Parámetro	Tipo	Descripción

`--enable-metrics`	Booleano	Permite recopilar métricas para generar perfiles de trabajo en la ejecución de trabajos. La opción predeterminada es 'falso'.
		
		Ejemplo:
		
		`%%configure`
		`{`
		`  "--enable-metrics": "true"`
		`}`

`--enable-continuous-cloudwatch-log`	Booleano	Permite un registro continuo en tiempo real de los trabajos de AWS Glue. La opción predeterminada es 'falso'.
		
		Ejemplo:
		
		`%%configure`
		`{`
		`  "--enable-continuous-cloudwatch-log": "true"`
		`}`

Parámetro	Tipo	Descripción

`--enable-continuous-log-filter`	Booleano	Especifica un filtro estándar o ningún filtro al crear o editar un trabajo habilitado para el registro continuo. El valor predeterminado es “verdadero”.
		
		Ejemplo:
		
		`%%configure`
		`{`
		`  "--enable-continuous-log-filter": "true"`
		`}`

`--continuous-log-stream-prefix`	Cadena	Especifica un prefijo de flujo de Amazon CloudWatch registro personalizado para un trabajo habilitado para el registro continuo. El valor predeterminado es nulo.
		
		Ejemplo:
		
		`%%configure`
		`{`
		`  "--continuous-log-stream-prefix": "prefix"`
		`}`

Parámetro	Tipo	Descripción

`--continuous-log-conversionPattern`	Cadena	Especifica un patrón de registro de conversión personalizado para un trabajo habilitado para el registro continuo. El valor predeterminado es nulo.
		
		Ejemplo:
		
		`%%configure`
		`{`
		`  "--continuous-log-conversionPattern": "pattern"`
		`}`

`--conf`	Cadena	Controla los parámetros de configuración de Spark. Es para casos de uso avanzados. Utilice `--conf` antes de cada parámetro. Ejemplo:
		
		`%%configure`
		`{`
		`    "--conf": "spark.hadoop.hive.metastore.glue.catalogid=123456789012 --conf hive.metastore.client.factory.class=com.amazonaws.glue.catalog.metastore.AWSGlueDataCatalogHiveClientFactory --conf hive.metastore.schema.verification=false"`
		`}`

Comandos mágicos de trabajos de Spark (ETL y streaming)

Nombre	Tipo	Descripción
`%worker_type`	Cadena	Estándar, G.1X o G.2X. También se debe configurar `number_of_workers` . El valor predeterminado de worker_type es G.1X.
`%connections`	Enumeración	Especifica una lista separada por comas de conexiones que se utilizarán en la sesión. Ejemplo: `%connections my_rds_connection dy_f = glue_context.create_dynamic _frame.from_catalog(databas e='rds_tables', table_nam e='sales_table')`
`%extra_py_files`	Enumeración	Lista separada por comas de archivos de Python adicionales de Simple Storage Service (Amazon S3).
`%extra_jars`	Enumeración	Lista separada por comas de archivos jar adicionales que se deben incluir en el clúster.
`%spark_conf`	Cadena	Especifique las configuraciones de Spark personalizadas para la sesión. Por ejemplo, `%spark_conf spark.ser ializer=org.apache.spark.se rializer.KryoSerializer` .

Comandos mágicos para trabajos de Ray

Nombre	Tipo	Descripción
`%min_workers`	Int	La cantidad mínima de trabajadores que se asignan a un trabajo de Ray. Valor predeterminado: 1. Ejemplo: `%min_workers 2`
`%object_memory_head`	Int	El porcentaje de memoria libre en el nodo de instancia después de un inicio en caliente. Mínimo: 0. Máximo: 100. Ejemplo: `%object_memory_head 100`
%object_memory_worker	Int	El porcentaje de memoria libre en los nodos de trabajo de la instancia después de un inicio en caliente. Mínimo: 0. Máximo: 100. Ejemplo: `%object_memory_worker 100`

Comando mágico de acción

Nombre	Tipo	Descripción
`%%sql`	Cadena	Ejecuta código SQL. Todas las líneas después del comando mágico inicial `%%sql` se pasarán como parte del código SQL. Ejemplo: `%%sql select * from rds_tables.sales_table`
`%matplot`	Figura de Matplotlib	Visualice sus datos con la biblioteca matplotlib.

Nombre	Tipo	Descripción

		Ejemplo:
		`import matplotlib.pyplot as plt`
		
		`# Set X-axis and Y-axis values`
		`x = [5, 2, 8, 4, 9]`
		`y = [10, 4, 8, 5, 2]`
		
		`# Create a bar chart`
		`plt.bar(x, y)`
		
		`# Show the plot`
		`%matplot plt`

`%plotly`	Figura de Plotly	Visualice sus datos con la biblioteca de plotly.
		
		Ejemplo:
		`import plotly.express as px`
		
		`#Create a graphical figure`
		`fig = px.line(x=["a","b","c"],`
		` y=[1,3,2], title="sample`
		` figure")`
		
		`#Show the figure`
		`%plotly fig`

Denominación de sesiones

AWS Gluelas sesiones interactivas son AWS recursos y requieren un nombre. Los nombres deben ser únicos para cada sesión y los administradores de IAM pueden restringirlos. Para obtener más información, consulte Sesiones interactivas con IAM. El kernel de Jupyter genera automáticamente nombres de sesión únicos para usted. No obstante, se puede poner nombre manualmente a las sesiones de dos formas:

1. Usando el archivo de AWS Command Line Interface configuración ubicado en`~.aws/config`. Consulte Configurar AWS Config con AWS Command Line Interface.
2. Utilizando los comandos mágicos `%session_id_prefix`. Consulte Comandos mágicos compatibles con las sesiones interactivas de AWS Glue para Jupyter.

Un nombre de sesión se genera de la siguiente manera:

- Cuando se proporcionan el prefijo y session_id: el nombre de la sesión será {prefijo}-{UUID}.
- Cuando no se proporciona nada: el nombre de la sesión será {UUID}.

Poner un prefijo a los nombres de las sesiones le permite reconocer su sesión al incluirla en la consola AWS CLI o.

Especificación de un rol de IAM para las sesiones interactivas

Debe especificar una función de AWS Identity and Access Management (IAM) para usarla con el código AWS Glue ETL que ejecuta en las sesiones interactivas.

El rol requiere los mismos permisos de IAM que los necesarios para ejecutar trabajos de AWS Glue. Consulte Crear un rol de IAM para AWS Glue para obtener más información sobre cómo crear un rol para trabajos y sesiones interactivas de AWS Glue.

Los roles de IAM se pueden especificar de dos formas:

- Utilice el archivo de AWS Command Line Interface configuración que se encuentra en `~.aws/config` (recomendado). Para obtener más información, consulte Configuración de sesiones con ~/.aws/config .

  > **Note**
  >
  > Cuando se utiliza el comando mágico `%profile`, se aplica la configuración para `glue_iam_role` de ese perfil.

- Utilizando el comando mágico %iam_role. Para obtener más información, consulte Comandos mágicos compatibles con las sesiones interactivas de AWS Glue para Jupyter.

Configuración de sesiones con perfiles con nombre

AWS Gluelas sesiones interactivas utilizan las mismas credenciales que el AWS Command Line Interface o boto3, y las sesiones interactivas respetan y funcionan con perfiles con nombre, como los que AWS CLI se encuentran en `~/.aws/config` (Linux y macOS) o `%USERPROFILE%\.aws\config` (Windows). Para obtener más información, consulte Uso de perfiles con nombre.

Las sesiones interactivas aprovechan los perfiles con nombre permitiendo que se especifiquen el rol de servicio y el prefijo de ID de sesión de AWS Glue en un perfil. Para configurar un rol de perfil, agregue una línea para la clave de `iam_role` o `session_id_prefix` al perfil con nombre tal y como se muestra a continuación. El `session_id_prefix` no requiere comillas. Por ejemplo, si desea agregar un `session_id_prefix`, ingrese el valor del `session_id_prefix=myprefix`.

```
[default]
region=us-east-1
aws_access_key_id=AKIAIOSFODNN7EXAMPLE
aws_secret_access_key=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
glue_iam_role=arn:aws:iam::<AccountID>:role/<GlueServiceRole>
session_id_prefix=<prefix_for_session_names>

[user1]
region=eu-west-1
aws_access_key_id=AKIAI44QH8DHBEXAMPLE
aws_secret_access_key=je7MtGbClwBF/2Zp9Utk/h3yCo8nvbEXAMPLEKEY
glue_iam_role=arn:aws:iam::<AccountID>:role/<GlueServiceRoleUser1>
session_id_prefix=<prefix_for_session_names_for_user1>
```

Si tiene un método personalizado para generar credenciales, también puede configurar el perfil para que utilice el parámetro `credential_process` en el archivo `~/.aws/config`. Por ejemplo:

```
[profile developer]
region=us-east-1
credential_process = "/Users/Dave/generate_my_credentials.sh" --username helen
```

Puede encontrar más información sobre cómo obtener credenciales a través del parámetro `credential_process` aquí: Obtención de credenciales con un proceso externo.

Si el perfil que está utilizando no tiene configurados una región o un `iam_role`, debe especificarlos mediante los comandos mágicos `%region` y `%iam_role` en la primera celda que ejecute.

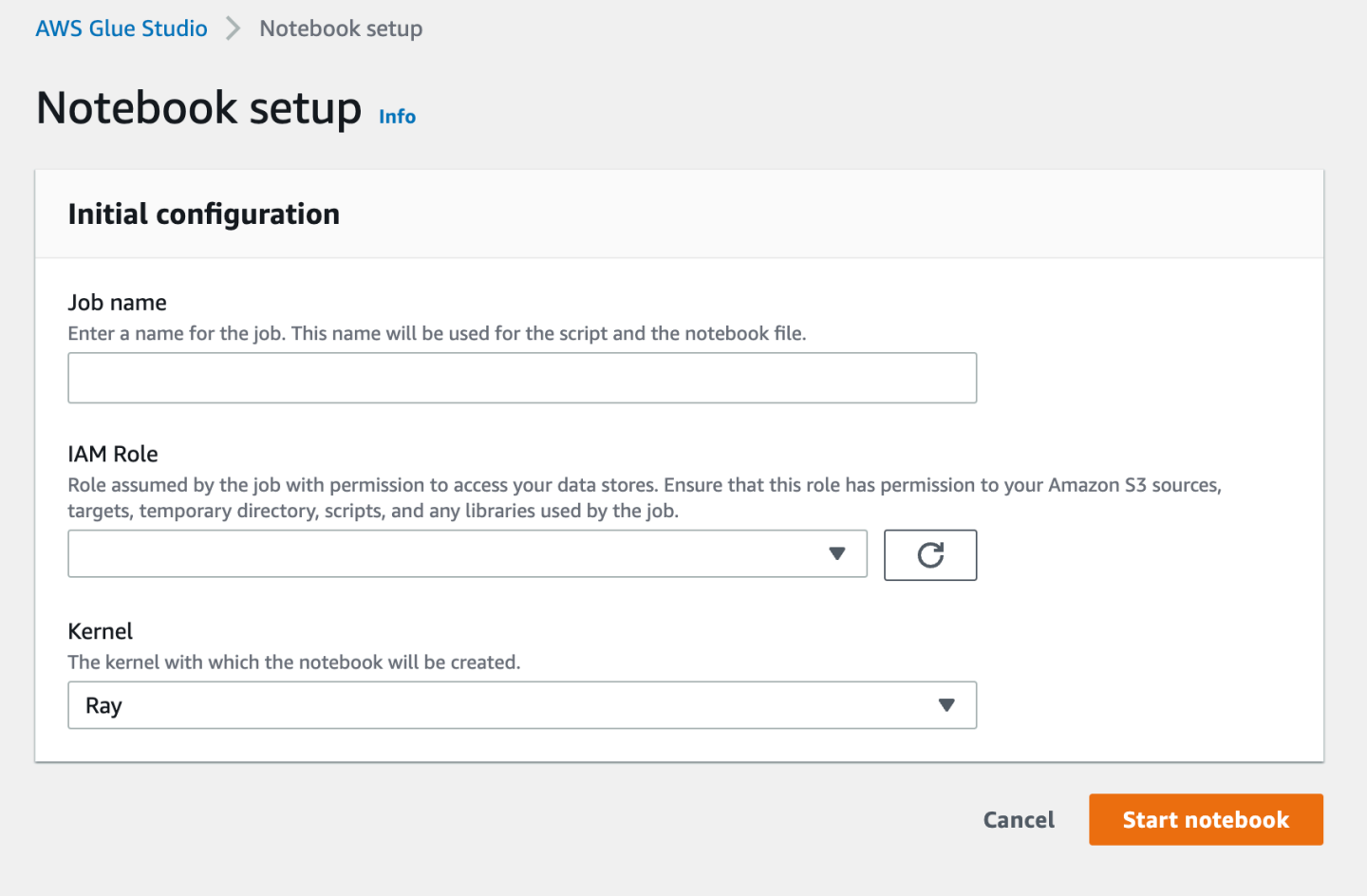
Introducción a las sesiones interactivas de AWS Glue For Ray (versión preliminar)

> **Warning**
>
> La vista previa de las sesiones interactivas de AWS Glue For Ray finalizó el 30 de abril de 2024. Ya no podrás crear nuevas sesiones interactivas AWS Glue para Ray.

> **Note**
>
> AWS Glue for Ray está disponible en EE.UU. Este (Norte de Virginia), EE.UU. Este (Ohio), EE.UU. Oeste (Oregón), Asia-Pacífico (Tokio) y Europa (Irlanda).

Sesiones interactivas de Ray en la AWS Glue Studio consola

En la página de trabajos de la AWS Glue Studio consola, selecciona la opción Jupyter Notebook existente. Se abrirá una página Notebook setup (Configuración del cuaderno) donde podrá seleccionar su kernel. Seleccione el kernel de Ray para iniciar una sesión interactiva de Ray. Para más información sobre las sesiones interactivas y cómo se utilizan, consulte the section called “Introducción a las sesiones interactivas de AWS Glue”.



Sesiones interactivas de Ray mediante el uso del kernel de Jupyter

Para usar el kernel de Ray fuera de la AWS Glue Studio consola, necesitará instalar el `aws-glue-sessions` paquete, que publicamos en PyPI. Para más información sobre cómo usar el paquete del kernel, consulte la documentación de the section called “Introducción a las sesiones interactivas de AWS Glue”.

Para actualizar o instalar el kernel, ejecute `pip install --upgrade aws-glue-sessions`. Necesitará la versión .37+ para usar el kernel de Ray.

Las sesiones interactivas de Ray tienen acceso a las mismas bibliotecas y versiones de Ray que los trabajos de Ray. En la versión preliminar, todas las sesiones interactivas de Ray utilizarán Ray 2.4.0.

Valores predeterminados del tiempo de espera de la sesión interactiva de Ray

- Tiempo de espera predeterminado (por sesión): 8 horas.

- Tiempo de espera de inactividad predeterminado: 1 hora.

Comandos mágicos compatibles con las sesiones interactivas de AWS Glue Ray

Los comandos mágicos del kernel de Jupyter de AWS Glue, cuando potencian las sesiones interactivas de Ray, son similares a los de las sesiones de Spark. Como referencia, consulte the section called " Configuración de las sesiones interactivas de AWS Glue para cuadernos de Jupyter y AWS Glue Studio".

Comandos mágicos de las sesiones

Los comandos mágicos de las sesiones son prácticamente los mismos que antes de la versión preliminar de AWS Glue para Ray. Para obtener más información sobre los comandos mágicos de sesión fuera de esta vista previa, consulte the section called "Comandos mágicos compatibles con las sesiones interactivas de AWS Glue para Jupyter". Presentamos un nuevo comando mágico para configurar el tipo de sesión en AWS Glue para Ray.

Nombre	Tipo	Descripción
`%glue_ray`	Cadena	Cambia el tipo de sesión a AWS Glue para Ray.

Comandos mágicos de configuración de AWS Glue

Los comandos mágicos para configurar AWS Glue en una sesión interactiva pueden ser diferente entre los tipos de sesión. Actualmente, solo se admite este subconjunto de comandos mágicos existentes cuando se usa AWS Glue para Ray.

> **Warning**
>
> Problema conocido: **`additional_python_modules`**
> En la vista previa de las sesiones interactivas de Ray, el uso del comando mágico `additional_python_modules` provocará problemas al guardar el cuaderno. Para configurar los módulos de Python para las sesiones de Ray, utilice el comando mágico `%%configure` para establecer el parámetro `pip-install` definido en the section called "Parámetros de los trabajos de Ray".

Nombre	Tipo	Descripción
`%%configure`	Diccionario	Especifica un diccionario en formato JSON que consta de todos los parámetros de configuración para una sesión. Cada parámetro se puede especificar aquí o mediante comandos mágicos individuales.
`%iam_role`	Cadena	Especifica un ARN de rol de IAM con el que ejecutar la sesión. Valor predeterm inado de ~/.aws/configure
`%number_of_workers`	int	Número de procesos de empleados de un worker_type definido que se asignan cuando se ejecuta un trabajo. También se debe configurar `worker_type` .
`%worker_type`	Cadena	En la versión preliminar de AWS Glue para Ray, el único tipo de proceso de trabajo admitido es Z.2X.
`%additional_python _modules`	Enumeración	Lista separada por comas de módulos de Python adicionales que se deben incluir en el clúster (pueden ser de Pypi o S3).

Comando mágico de acción

Las sesiones de AWS Glue Ray no admiten ningún tipo de comando mágico de acción.

Sesiones interactivas con IAM

En estas secciones, se describen las consideraciones de seguridad para sesiones interactivas de AWS Glue.

Temas

- Entidades principales de IAM que se utilizan con sesiones interactivas
- Configuración de una entidad principal del cliente

- Configuración de un rol de tiempo de ejecución
- Haga que su sesión sea privada con TagOnCreate
- Consideraciones respecto de la política de IAM

Entidades principales de IAM que se utilizan con sesiones interactivas

Se utilizan dos entidades principales de IAM con sesiones interactivas de AWS Glue.

- Entidad principal del cliente: la entidad principal del cliente (ya sea un usuario o un rol) autoriza las operaciones de la API para sesiones interactivas de un cliente de AWS Glue configurado con las credenciales basadas en la identidad de la entidad principal. Por ejemplo, esta podría ser un rol de IAM que suele utilizarse para acceder a la consola de AWS Glue. También podría ser una función asignada a un usuario de IAM cuyas credenciales se utilizan para el núcleo de Jupyter de las AWS Command Line Interface sesiones interactivas o a un AWS Glue cliente utilizado por las sesiones interactivas.
- Runtime role (Rol de tiempo de ejecución): el rol de tiempo de ejecución es un rol de IAM que la entidad principal del cliente pasa a las operaciones de la API de las sesiones interactivas. AWS Glue utiliza este rol para ejecutar instrucciones en la sesión. Por ejemplo, este rol podría ser el que se utiliza para ejecutar trabajos de ETL de AWS Glue.

  Para obtener más información, consulte Configuración de un rol de tiempo de ejecución.

Configuración de una entidad principal del cliente

Debe adjuntar una política de identidad a la entidad principal del cliente para permitirle llamar a la API de sesiones interactivas. Este rol debe tener el acceso `iam:PassRole` al rol de ejecución que se pasaría a la API de sesiones interactivas, como `CreateSession`. Por ejemplo, puedes adjuntar la política AWSGlueConsoleFullAccessgestionada a un rol de IAM que permita a los usuarios de tu cuenta con la política asociada acceder a todas las sesiones creadas en tu cuenta (como el extracto de tiempo de ejecución o el extracto de cancelación).

Si desea proteger su sesión y convertirla en privada solo para determinadas funciones de IAM, como las asociadas al usuario que creó la sesión, puede utilizar el denominado Control de autorización basado en etiquetas de AWS Glue Interactive Session. TagOnCreate Para obtener más información, consulte Haga que su sesión sea privada con TagOnCreate cómo una política gestionada con alcance limitado y basada en etiquetas de propietario puede hacer que su sesión sea privada.

TagOnCreate Para obtener más información sobre las políticas basadas en la identidad, consulte Políticas basadas en la identidad para AWS Glue.

Configuración de un rol de tiempo de ejecución

Debe transferir una función de IAM a la operación de la CreateSession API para poder asumir y ejecutar declaraciones en las sesiones interactivas. AWS Glue El rol debe tener los mismos permisos de IAM que los necesarios para ejecutar trabajos típicos de AWS Glue. Por ejemplo, puede crear un rol de servicio mediante la AWSGlueServiceRolepolítica que permite llamar AWS Glue a AWS los servicios en su nombre. Si utiliza la consola de AWS Glue, se creará automáticamente un rol de servicio en su nombre o se utilizará uno existente. También puede crear su propio rol de IAM y adjuntar su propia política de IAM para conceder permisos similares.

Si desea proteger su sesión y hacerla privada solo para el usuario que la creó, puede utilizar el denominado Control de autorización basado en etiquetas de AWS Glue Interactive Session TagOnCreate. Para obtener más información, consulte Haga que su sesión sea privada con TagOnCreate cómo una política gestionada con alcance limitado y basada en etiquetas de propietario puede hacer que su sesión sea privada con. TagOnCreate Para obtener más información sobre políticas basadas en identidad, consulte Políticas basadas en la identidad para Glue AWS. Si va a crear el rol de ejecución usted mismo desde la consola de IAM y desea que su servicio sea privado con la TagOnCreate función, siga los pasos que se indican a continuación.

1. Cree un rol de IAM con el tipo de rol configurado como `Glue`.
2. Adjunta esta política AWS Glue gestionada: AwsGlueSessionUserRestrictedServiceRole
3. Añada el nombre de la política al nombre de la función como prefijo. AwsGlueSessionUserRestrictedServiceRole Por ejemplo, puede crear un rol con el nombre AwsGlueSessionUserRestrictedServiceRole-myrole y adjuntar AWS Glue una política administrada. AwsGlueSessionUserRestrictedServiceRole
4. Adjunte una política de confianza como la siguiente para permitir que AWS Glue asuma el rol:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "glue.amazonaws.com"
        ]
```

```
      },
      "Action": [
        "sts:AssumeRole"
      ]
    }
  ]
}
```

Para el kernel de Jupyter de una sesión interactiva, puede especificar la `iam_role` clave en su perfil. AWS Command Line Interface Para obtener más información, consulte Configuración de sesiones con ~/.aws/config . Si interactúa con sesiones interactivas mediante un cuaderno de AWS Glue, puede transferir el rol de ejecución en el comando mágico `%iam_role` en la primera celda que ejecute.

Haga que su sesión sea privada con TagOnCreate

Las sesiones interactivas de AWS Glue son compatibles con el etiquetado y la autorización basada en etiquetas (TBAC) de las sesiones interactivas como recurso con nombre. Además del uso del TBAC TagResource y UntagResource las API, las sesiones AWS Glue interactivas admiten la TagOnCreate función de «etiquetar» una sesión con una etiqueta determinada solo durante la creación de la sesión con CreateSession la operación. Esto también significa que esas etiquetas se eliminarán el DeleteSession, también conocido como. UntagOnDelete

TagOnCreate ofrece un potente mecanismo de seguridad para que su sesión sea privada para el creador de la sesión. Por ejemplo, puede adjuntar una política de IAM con «propietario» RequestTag y un valor de $ {AWS:userID} al cliente principal (como un usuario) para permitir la creación de una sesión solo si en la solicitud se proporciona una etiqueta de «propietario» con un valor coincidente con el ID de usuario de la persona que llama. CreateSession Esta política permite que las sesiones interactivas de AWS Glue puedan crear un recurso de sesión y etiquetar la sesión con la etiqueta userId solo durante la creación de la sesión. Además, puedes limitar el acceso (por ejemplo, a la ejecución de declaraciones) a tu sesión únicamente al creador (también conocido como etiqueta de propietario con el valor $ {aws:userID}) de la sesión adjuntando una política de IAM con la palabra «propietario» ResourceTag al rol de ejecución que hayas transferido durante el proceso CreateSession.

Con el fin de facilitar el uso de la TagOnCreate función para hacer que una sesión sea privada para el creador de la sesión, AWS Glue proporciona políticas gestionadas y funciones de servicio especializadas.

Si desea crear una sesión AWS Glue interactiva utilizando un elemento AssumeRole principal de IAM (es decir, utilizando una credencial vendida asumiendo un rol de IAM) y quiere que la sesión sea privada para el creador, utilice políticas similares a las y, respectivamente. AWSGlueSessionUserRestrictedNotebookPolicyAWSGlueSessionUserRestrictedNotebookServiceRole Estas políticas permiten AWS Glue usar $ {aws:PrincipalTag} para extraer el valor de la etiqueta de propietario. Esto requiere que pases una etiqueta de ID de usuario con el valor $ {aws:UserID} como en la credencial del supuesto rol. SessionTag Consulte Etiquetas de sesión de ID. Si utiliza una instancia de Amazon EC2 con un perfil de instancia que vende la credencial y desea crear una sesión o interactuar con la sesión desde la instancia de Amazon EC2, tendrá que pasar una etiqueta de ID de usuario con el valor $ {aws:userId} como credencial de asunción de rol. SessionTag

Por ejemplo, si va a crear una sesión con una credencial AssumeRole principal de IAM y desea que su servicio sea privado con la TagOnCreate función, siga los pasos que se indican a continuación.

1. Cree un rol de tiempo de ejecución usted mismo desde la consola de IAM. Adjunta esta política AWS Glue gestionada AwsGlueSessionUserRestrictedNotebookServiceRoley añade el nombre de la política al nombre de la función como prefijo. AwsGlueSessionUserRestrictedNotebookServiceRole Por ejemplo, puede crear un rol con el nombre AwsGlueSessionUserRestrictedNotebookServiceRole-myrole y adjuntar AWS Glue una política administrada. AwsGlueSessionUserRestrictedNotebookServiceRole
2. Adjunte una política de confianza como la que aparece a continuación para permitir que AWS Glue asuma el rol anterior.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "glue.amazonaws.com"
        ]
      },
      "Action": [
        "sts:AssumeRole"
       ]
    }
   ]
}
```

3. Cree otro rol con un prefijo AwsGlueSessionUserRestrictedNotebookPolicyy adjunte la política AWS Glue administrada AwsGlueSessionUserRestrictedNotebookPolicypara que la sesión sea privada. Además de la política gestionada, adjunte la siguiente política en línea para permitir iam: PassRole al rol que creó en el paso 1.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/
AwsGlueSessionUserRestrictedNotebookServiceRole*"
            ],
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": [
                        "glue.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

4. Adjunte una política de confianza como la siguiente a la IAM de AWS Glue para asumir el rol.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Principal": {
            "Service": [
                "glue.amazonaws.com"
            ]
        },
        "Action": [
```

```
            "sts:AssumeRole",
            "sts:TagSession"
        ]
    }]
}
```

> **Note**
>
> Si lo desea, puede utilizar un único rol (por ejemplo, el rol de bloc de notas) y adjuntar las dos políticas AwsGlueSessionUserRestrictedNotebookServiceRolegestionadas anteriores y. AwsGlueSessionUserRestrictedNotebookPolicy Adjunte también la política insertada adicional para permitir `iam:passrole` del rol a AWS Glue. Y, finalmente, adjunte la política de confianza anterior para permitir `sts:AssumeRole` y `sts:TagSession`.

AWSGlueSessionUserRestrictedNotebookPolicy

Solo AWSGlueSessionUserRestrictedNotebookPolicy permite crear una sesión AWS Glue interactiva desde un bloc de notas si la clave «propietario» y un valor coinciden con el AWS seudónimo del principal (usuario o rol). Para más información, consulte Dónde puede utilizar variables de política. Esta política se adjunta a la entidad principal (usuario o rol) que crea cuadernos de sesión interactiva de AWS Glue desde AWS Glue Studio. Esta política también permite un acceso suficiente al AWS Glue Studio cuaderno para interactuar con los recursos de la sesión AWS Glue Studio interactiva que se crean con el valor de la etiqueta «propietario» que coincide con el ID de AWS usuario del director. Esta política deniega el permiso para cambiar o eliminar la etiqueta de “propietario” de un recurso de sesión de AWS Glue después de crear la sesión.

AWSGlueSessionUserRestrictedNotebookServiceRole

Esto AWSGlueSessionUserRestrictedNotebookServiceRoleproporciona acceso suficiente al AWS Glue Studio cuaderno para interactuar con los recursos de la sesión AWS Glue interactiva que se crean con el valor de la etiqueta «propietario» que coincide con el seudónimo del AWS usuario principal (usuario o rol) del creador del cuaderno. Para más información, consulte Dónde puede utilizar variables de política. Esta política de rol de servicio está asociada al rol que se transfiere por arte de magia a un bloc de notas o que se transfiere como rol de ejecución a la CreateSession API. Esta política también permite crear una sesión AWS Glue interactiva a partir de un bloc de

notas únicamente si la clave «propietario» y un valor coinciden con el seudónimo del AWS usuario principal. Esta política deniega el permiso para cambiar o eliminar la etiqueta de “propietario” de un recurso de sesión de AWS Glue después de crear la sesión. Esta política también incluye permisos para escribir y leer desde buckets de Amazon S3, escribir CloudWatch registros y crear y eliminar etiquetas para los recursos de Amazon EC2 utilizados por. AWS Glue

Convertir la sesión en privada con políticas de usuarios

Puede adjuntar las AWSGlueSessionUserRestrictedPolicyfunciones de IAM asociadas a cada uno de los usuarios de su cuenta para evitar que creen una sesión únicamente con una etiqueta de propietario cuyo valor coincida con su propio ${aws:userId}. En lugar de utilizar las AWSGlueSessionUserRestrictedNotebookPolicyy, AWSGlueSessionUserRestrictedNotebookServiceRoledebe utilizar políticas similares a las y, respectivamente. AWSGlueSessionUserRestrictedPolicyAWSGlueSessionUserRestrictedServiceRole Para obtener más información, consulte Uso de políticas basadas en identidad. Esta política reduce el ámbito de acceso a una sesión solo al creador, también conocido como ${aws:userId} del usuario que creó la sesión con la etiqueta de propietario que contiene su propio ${aws:userId}. Si ha creado el rol de ejecución usted mismo mediante la consola de IAM siguiendo los pasos descritos anteriormenteConfiguración de un rol de tiempo de ejecución, además de adjuntar la política AwsGlueSessionUserRestrictedPolicygestionada, adjunte también la siguiente política integrada a cada uno de los usuarios de su cuenta `iam:PassRole` para permitir el rol de ejecución que creó anteriormente.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "iam:PassRole"
        ],
        "Resource": [
            "arn:aws:iam::*:role/AwsGlueSessionUserRestrictedServiceRole*"
        ],
        "Condition": {
            "StringLike": {
                "iam:PassedToService": [
                    "glue.amazonaws.com"
                ]
            }
```

```
        }
    }]
}
```

AWSGlueSessionUserRestrictedPolicy

Solo permite crear una sesión AWS Glue interactiva mediante la CreateSession API si se AWSGlueSessionUserRestrictedPolicyproporciona una clave de etiqueta («propietario») y un valor que coincidan con su ID AWS de usuario. Esta política de identidad se adjunta al usuario que invoca la CreateSession API. Esta política también permite interactuar con los recursos de la sesión AWS Glue interactiva que se crearon con una etiqueta de «propietario» y un valor que coincidan con su identificador AWS de usuario. Esta política deniega el permiso para cambiar o eliminar la etiqueta de "propietario" de un recurso de sesión de AWS Glue después de crear la sesión.

AWSGlueSessionUserRestrictedServiceRole

AWSGlueSessionUserRestrictedServiceRoleProporciona acceso completo a todos los AWS Glue recursos, excepto a las sesiones, y permite a los usuarios crear y usar solo las sesiones interactivas asociadas al usuario. Esta política también incluye otros permisos necesarios AWS Glue para gestionar los recursos de Glue en otros AWS servicios. La política también permite añadir etiquetas a AWS Glue los recursos de otros AWS servicios.

Consideraciones respecto de la política de IAM

Las sesiones interactivas son recursos de IAM en AWS Glue. Debido a que son recursos de IAM, el acceso a una sesión y la interacción con la misma se rigen por las políticas de IAM. Según las políticas de IAM adjuntadas a una entidad principal del cliente o rol de ejecución configurado por un administrador, una entidad principal del cliente (usuario o rol) podrá crear nuevas sesiones e interactuar con las suyas y con otras.

Si un administrador ha incorporado una política de IAM AWSGlueServiceRole que permita el acceso a todos AWS Glue los recursos de esa cuenta, el cliente principal podrá colaborar entre sí. AWSGlueConsoleFullAccess Por ejemplo, un usuario podrá interactuar con las sesiones creadas por otros usuarios si las políticas lo permiten.

Si desea configurar una política que se adapte a sus necesidades específicas, consulte la documentación de IAM sobre la configuración de los recursos de una política. Por ejemplo, para aislar las sesiones que pertenecen a un usuario, puede utilizar la TagOnCreate función compatible con las sesiones AWS Glue interactivas. Consulte Haga que su sesión sea privada con TagOnCreate .

Las sesiones interactivas permiten limitar la creación de sesiones en función de determinadas condiciones de la VPC. Consulte Políticas que controlan la configuración mediante claves de condición.

Conversión de un script o cuaderno en un trabajo de AWS Glue

Hay dos maneras de convertir un script o cuaderno en un trabajo de AWS Glue:

- Utilice nbconvert para convertir el archivo del documento de cuaderno `.ipynb` de Jupyter en un archivo `.py`. Para obtener más información, consulte nbconvert: convertir cuadernos a otros formatos.
- Cargue el archivo en cuadernos de AWS Glue Studio.
  - En la consola de AWS Glue Studio, elija Jobs (Trabajos) en el menú de navegación izquierdo.
  - En la sección Create job (Crear trabajo), elija Jupyter Notebook.
  - En la sección Options (Opciones), elija Upload and edit an existing notebook (Cargar y editar un cuaderno existente).
  - Seleccione Choose file (Elegir archivo) para cargar un archivo `.ipynb`.

Sesiones interactivas de AWS Glue para streaming

Cambio del tipo de sesión de streaming

Utilice el comando mágico de configuración de las sesiones interactivas de AWS Glue, `%streaming`, para definir el trabajo que vaya a ejecutar e inicializar una sesión interactiva de streaming.

Muestreo de flujo de entrada para desarrollo interactivo

Una herramienta que hemos creado para ayudar a mejorar la experiencia AWS Glue interactiva en las sesiones interactivas es la adición de un nuevo método `GlueContext` para obtener una instantánea de una transmisión en formato estático DynamicFrame. `GlueContext`le permite inspeccionar, interactuar e implementar su flujo de trabajo.

Con la instancia de la clase `GlueContext`, podrá localizar el método `getSampleStreamingDynamicFrame`. Los argumentos requeridos para este método son:

- `dataFrame`: The Spark Streaming DataFrame

- `options`: consulte las opciones disponibles a continuación

Entre las opciones disponibles se incluyen:

- windowSize: también se denomina duración de microlote. Este parámetro determinará cuánto tiempo esperará una consulta de streaming después de que se haya desencadenado el lote anterior. El valor de este parámetro debe ser inferior al de `pollingTimeInMs`.
- pollingTimeInSra.: El tiempo total durante el que se ejecutará el método. Desencadenará al menos un microlote para obtener registros de muestra del flujo de entrada.
- recordPollingLimit: Este parámetro le ayuda a limitar el número total de registros de la transmisión que se van a sondear.
- (Opcional) También se puede utilizar `writeStreamFunction` para aplicar esta función personalizada a cada función de muestreo de registros. Consulte los ejemplos en Scala y Python que aparecen a continuación.

Scala

```
val sampleBatchFunction = (batchDF: DataFrame, batchId: Long) => {//Optional but
 you can replace your own forEachBatch function here}
val jsonString: String = s"""{"pollingTimeInMs": "10000", "windowSize": "5
 seconds"}"""
val dynFrame = glueContext.getSampleStreamingDynamicFrame(YOUR_STREAMING_DF,
 JsonOptions(jsonString), sampleBatchFunction)
dynFrame.show()
```

Python

```
def sample_batch_function(batch_df, batch_id):
       //Optional but you can replace your own forEachBatch function here
options = {
            "pollingTimeInMs": "10000",
            "windowSize": "5 seconds",
        }
glue_context.getSampleStreamingDynamicFrame(YOUR_STREAMING_DF, options,
 sample_batch_function)
```

> **Note**
>
> Cuando el elemento `DynFrame` que se muestrea está vacío, puede deberse a varios motivos:
>
> - El origen del streaming está configurado como “Latest” (Más reciente) y no se han ingerido datos nuevos durante el período de muestra.
> - El tiempo de sondeo no es suficiente para procesar los registros que ha ingerido. No aparecerán datos a menos que se haya procesado todo el lote.

Ejecución de aplicaciones de streaming en las sesiones interactivas

En las sesiones interactivas de AWS Glue, se puede ejecutar una aplicación de streaming de AWS Glue del mismo modo que se crearía una aplicación de streaming en la consola de AWS Glue. Dado que las sesiones interactivas se basan en sesiones, si se producen excepciones en el motor de ejecución no se detiene la sesión. Ahora ofrecemos el beneficio adicional de desarrollar una función por lotes de manera iterativa. Por ejemplo:

```
def batch_function(data_frame, batch_id):
    log.info(data_frame.count())
    invalid_method_call()
glueContext.forEachBatch(frame=streaming_df, batch_function = batch_function, options =
 {**})
```

En el ejemplo anterior, incluimos un uso no válido de un método y, a diferencia de trabajos de AWS Glue normales, que cerrarán toda la aplicación, el contexto de codificación y las definiciones del usuario se conservan por completo y la sesión sigue funcionando. No es necesario arrancar un nuevo clúster y volver a ejecutar toda la transformación anterior. Eso permite centrarse en iterar rápidamente las implementaciones de una función por lotes para obtener los resultados deseados.

Es importante tener en cuenta que Interactive Sessions evalúa cada instrucción con una perspectiva de bloqueo, para que la sesión solo ejecute una instrucción a la vez. Dado que las consultas de streaming son continuas y nunca terminan, las sesiones con consultas de streaming activas no podrán gestionar ninguna instrucción de seguimiento a menos que se interrumpan. Puede ejecutar el comando de interrupción directamente desde Jupyter Notebook y nuestro kernel se encargará de realizar la cancelación.

Tomemos como ejemplo la siguiente secuencia de instrucciones que están esperando su ejecución:

```
Statement 1:
      val number = df.count()
      #Spark Action with deterministic result
      Result: 5

Statement 2:
      streamingQuery.start().awaitTermination()
      #Spark Streaming Query that will be executing continously
      Result: Constantly updated with each microbatch

Statement 3:
      val number2 = df.count()
      #This will not be executed as previous statement will be running indefinitely
```

Desarrollo y prueba de scripts de trabajos de AWS Glue a nivel local

Cuando desarrolle y pruebe sus scripts de trabajo Spark de AWS Glue, hay varias opciones disponibles:

- Consola de AWS Glue Studio
  - Visual editor (Editor visual)
  - Editor de scripts
  - Cuaderno de AWS Glue Studio
- Sesiones interactivas
  - Cuadernos de Jupyter
- Imagen de Docker
  - Desarrollo local
  - Desarrollo remoto
- Biblioteca de ETL AWS Glue Studio
  - Desarrollo local

Puede elegir cualquiera de las opciones anteriores en función de sus requisitos.

Si prefiere no tener código o menos experiencia en código, el editor visual de AWS Glue Studio es una buena opción.

Si prefiere una experiencia de cuaderno interactiva, el cuaderno de AWS Glue Studio es una buena opción. Para obtener más información, consulte Uso de los cuadernos con AWS Glue Studio y AWS Glue. Si desea utilizar su propio entorno local, las sesiones interactivas son una buena opción. Para obtener más información, consulte Uso de sesiones interactivas con AWS Glue.

Si prefiere la experiencia de desarrollo local o remoto, la imagen de Docker es una buena opción. Esto le ayuda a desarrollar y probar el script de trabajo Spark de AWS Glue en cualquier lugar que prefiera sin tener que incurrir en costos de AWS Glue.

Si prefiere el desarrollo local sin Docker, instalar el directorio de bibliotecas ETL de AWS Glue en forma local es una buena opción.

Desarrollo mediante AWS Glue Studio

El editor visual AWS Glue Studio es una interfaz gráfica que facilita la creación, ejecución y supervisión de los trabajos de extracción, transformación y carga (ETL) en AWS Glue. Puede componer visualmente flujos de trabajo de transformación de datos y ejecutarlos sin problemas en el motor de ETL sin servidor basado en Apache Spark de AWS Glue. Puede inspeccionar los resultados de esquema y datos en cada paso del trabajo. Para más información, consulte la Guía del usuario de AWS Glue Studio.

Desarrollo mediante sesiones interactivas

Las sesiones interactivas le permiten crear y probar aplicaciones desde el entorno que elija. Para obtener más información, consulte Uso de sesiones interactivas con AWS Glue.

Desarrollo mediante una imagen de Docker

> **Note**
>
> No se han realizado pruebas de las instrucciones de esta sección en los sistemas operativos Microsoft Windows.

Para el desarrollo local y las pruebas en plataformas Windows, consulte el blog Building an AWS Glue ETL pipeline locally without an AWS account (Creación de una canalización de ETL de AWS Glue en forma local sin una cuenta AWS).

Para una plataforma de datos lista para producción, el proceso de desarrollo y la canalización de CI/CD para trabajos de AWS Glue es un tema clave. Puede desarrollar y probar trabajos de AWS Glue de manera flexible en un contenedor de Docker. AWS Glue aloja las imágenes de Docker en Docker Hub para configurar el entorno de desarrollo con utilidades adicionales. Puede utilizar su IDE, cuaderno o REPL preferidos mediante la biblioteca de ETL de AWS Glue. En este tema, se describe cómo desarrollar y probar trabajos de versión 4.0 AWS Glue en un contenedor de Docker mediante la utilización de una imagen de Docker.

Las siguientes imágenes de Docker están disponibles para AWS Glue en Docker Hub.

- Para la versión 4.0 AWS Glue: `amazon/aws-glue-libs:glue_libs_4.0.0_image_01`
- Para la versión 3.0 de AWS Glue: `amazon/aws-glue-libs:glue_libs_3.0.0_image_01`
- Para la versión 2.0 de AWS Glue: `amazon/aws-glue-libs:glue_libs_2.0.0_image_01`

Estas imágenes son para x86_64. Se recomienda realizar pruebas en esta arquitectura. Sin embargo, es posible rediseñar una solución de desarrollo local a partir de imágenes base no compatibles.

Este ejemplo describe el uso de `amazon/aws-glue-libs:glue_libs_4.0.0_image_01` y la ejecución del contenedor en un equipo local. Esta imagen de contenedor ha sido probada para trabajos de versión 3.3 de Spark de AWS Glue. Esta imagen contiene lo siguiente:

- Amazon Linux
- Biblioteca de ETL de AWS Glue (aws-glue-libs)
- Apache Spark 3.3.0
- Servidor de historial de Spark
- Laboratorio de Jupyter
- Livy
- Otras dependencias de biblioteca (el mismo conjunto que las del sistema de trabajo de AWS Glue)

Complete una de las siguientes secciones de acuerdo con sus requisitos:

- Configurar el contenedor para usar spark-submit
- Configurar el contenedor para usar el intérprete de comandos REPL (PySpark)
- Configurar el contenedor para utilizar Pytest
- Configurar el contenedor para usar Jupyter Lab
- Configurar el contenedor para utilizar Visual Studio Code

Requisitos previos

Antes de empezar, asegúrese de que Docker esté instalado y que el daemon Docker se esté ejecutando. Para ver las instrucciones de instalación, consulte la documentación de Docker para Mac o Linux. La máquina que ejecuta Docker aloja el contenedor de AWS Glue. Asegúrese también de tener al menos 7 GB de espacio en disco para la imagen del host que ejecuta Docker.

Para obtener más información sobre las restricciones en el momento de desarollar el código AWS Glue en forma local, consulte Restricciones de desarrollo local.

Configuración de AWS

Para habilitar llamadas a la API de AWS desde el contenedor, configure las credenciales de AWS siguiendo estos pasos. En las siguientes secciones, utilizaremos este perfil con nombre AWS.

1. Configure la AWS CLI, con un perfil con nombre. Para obtener más información sobre la configuración de AWS CLI, consulte Opciones de los archivos de configuración y credenciales en la documentación de AWS CLI.
2. Ejecute el siguiente comando en un terminal:

```
PROFILE_NAME="<your_profile_name>"
```

Es posible que también necesite configurar la variable de entorno AWS_REGION para especificar la Región de AWS a la cual se enviarán las solicitudes.

Configuración y ejecución del contenedor

La configuración del contenedor para que ejecute el código de PySpark a través del comando spark-submit incluye los siguientes pasos de alto nivel:

1. Para extraer la imagen de Docker Hub.

2. Ejecute el contenedor.

Extracción de la imagen desde Docker Hub

Ejecute el siguiente comando para extraer la imagen del Docker Hub:

```
docker pull amazon/aws-glue-libs:glue_libs_4.0.0_image_01
```

Ejecución del contenedor

Ahora puede ejecutar un contenedor mediante esta imagen. Puede elegir cualquiera de los siguientes según sus requisitos.

spark-submit

Puede ejecutar un script de trabajo de AWS Glue ejecutando el comando `spark-submit` en el contenedor.

1. Escriba el script y guárdelo como `sample1.py` en el directorio `/local_path_to_workspace`. El código de muestra se incluye como apéndice de este tema.

```
$ WORKSPACE_LOCATION=/local_path_to_workspace
$ SCRIPT_FILE_NAME=sample.py
$ mkdir -p ${WORKSPACE_LOCATION}/src
$ vim ${WORKSPACE_LOCATION}/src/${SCRIPT_FILE_NAME}
```

2. Ejecute el siguiente comando para ejecutar el comando `spark-submit` en el contenedor para enviar una nueva aplicación Spark:

```
$ docker run -it -v ~/.aws:/home/glue_user/.aws -v $WORKSPACE_LOCATION:/
home/glue_user/workspace/ -e AWS_PROFILE=$PROFILE_NAME -e DISABLE_SSL=true
 --rm -p 4040:4040 -p 18080:18080 --name glue_spark_submit amazon/aws-glue-
libs:glue_libs_4.0.0_image_01 spark-submit /home/glue_user/workspace/src/
$SCRIPT_FILE_NAME
...22/01/26 09:08:55 INFO DAGScheduler: Job 0 finished: fromRDD at
 DynamicFrame.scala:305, took 3.639886 s
root
|-- family_name: string
|-- name: string
|-- links: array
| |-- element: struct
| | |-- note: string
```

```
| | |-- url: string
|-- gender: string
|-- image: string
|-- identifiers: array
| |-- element: struct
| | |-- scheme: string
| | |-- identifier: string
|-- other_names: array
| |-- element: struct
| | |-- lang: string
| | |-- note: string
| | |-- name: string
|-- sort_name: string
|-- images: array
| |-- element: struct
| | |-- url: string
|-- given_name: string
|-- birth_date: string
|-- id: string
|-- contact_details: array
| |-- element: struct
| | |-- type: string
| | |-- value: string
|-- death_date: string

...
```

3. (Opcionalmente) Configure `spark-submit` para que se adapte a su entorno. Por ejemplo, puede transferir sus dependencias con la configuración de `--jars`. Para obtener más información, consulte Launching Applications with spark-submit en la documentación de Spark.

Intérprete de comandos REPL (Pyspark)

Puede ejecutar el intérprete de comandos REPL (bucles de read-eval-print) para desarrollo interactivo.

Ejecute el siguiente comando para ejecutar el comando PySpark en el contenedor e iniciar el intérprete de comandos REPL:

```
$ docker run -it -v ~/.aws:/home/glue_user/.aws -e AWS_PROFILE=$PROFILE_NAME -e
 DISABLE_SSL=true --rm -p 4040:4040 -p 18080:18080 --name glue_pyspark amazon/aws-glue-
libs:glue_libs_4.0.0_image_01 pyspark
```

```
...
 ____ __
 / __/__ ___ _____/ /__
 _\ \/ _ \/ _ `/ __/ '_/
 /__ / .__/\_,_/_/ /_/\_\  version 3.1.1-amzn-0
 /_/

Using Python version 3.7.10 (default, Jun 3 2021 00:02:01)
Spark context Web UI available at http://56e99d000c99:4040
Spark context available as 'sc' (master = local[*], app id = local-1643011860812).
SparkSession available as 'spark'.
>>>
```

Pytest

Para pruebas de unidad, puede utilizar pytest para scripts de trabajo Spark de AWS Glue.

Ejecute los comandos siguientes para la preparación.

```
$ WORKSPACE_LOCATION=/local_path_to_workspace
$ SCRIPT_FILE_NAME=sample.py
$ UNIT_TEST_FILE_NAME=test_sample.py
$ mkdir -p ${WORKSPACE_LOCATION}/tests
$ vim ${WORKSPACE_LOCATION}/tests/${UNIT_TEST_FILE_NAME}
```

Ejecute el siguiente comando para ejecutar `pytest` en el conjunto de pruebas:

```
$ docker run -it -v ~/.aws:/home/glue_user/.aws -v $WORKSPACE_LOCATION:/home/glue_user/
workspace/ -e AWS_PROFILE=$PROFILE_NAME -e DISABLE_SSL=true --rm -p 4040:4040 -p
 18080:18080 --name glue_pytest amazon/aws-glue-libs:glue_libs_4.0.0_image_01 -c
 "python3 -m pytest"
starting org.apache.spark.deploy.history.HistoryServer,
 logging to /home/glue_user/spark/logs/spark-glue_user-
org.apache.spark.deploy.history.HistoryServer-1-5168f209bd78.out
*============================================================= test session starts
 ==============================================================
*platform linux -- Python 3.7.10, pytest-6.2.3, py-1.11.0, pluggy-0.13.1
rootdir: /home/glue_user/workspace
plugins: anyio-3.4.0
*collected 1 item  *

tests/test_sample.py . [100%]
```

```
============================================================= warnings summary
 =============================================================
tests/test_sample.py::test_counts
 /home/glue_user/spark/python/pyspark/sql/context.py:79: DeprecationWarning: Deprecated
 in 3.0.0. Use SparkSession.builder.getOrCreate() instead.
 DeprecationWarning)

-- Docs: https://docs.pytest.org/en/stable/warnings.html
========================================================== 1 passed, *1 warning* in
 21.07s ===========================================================
```

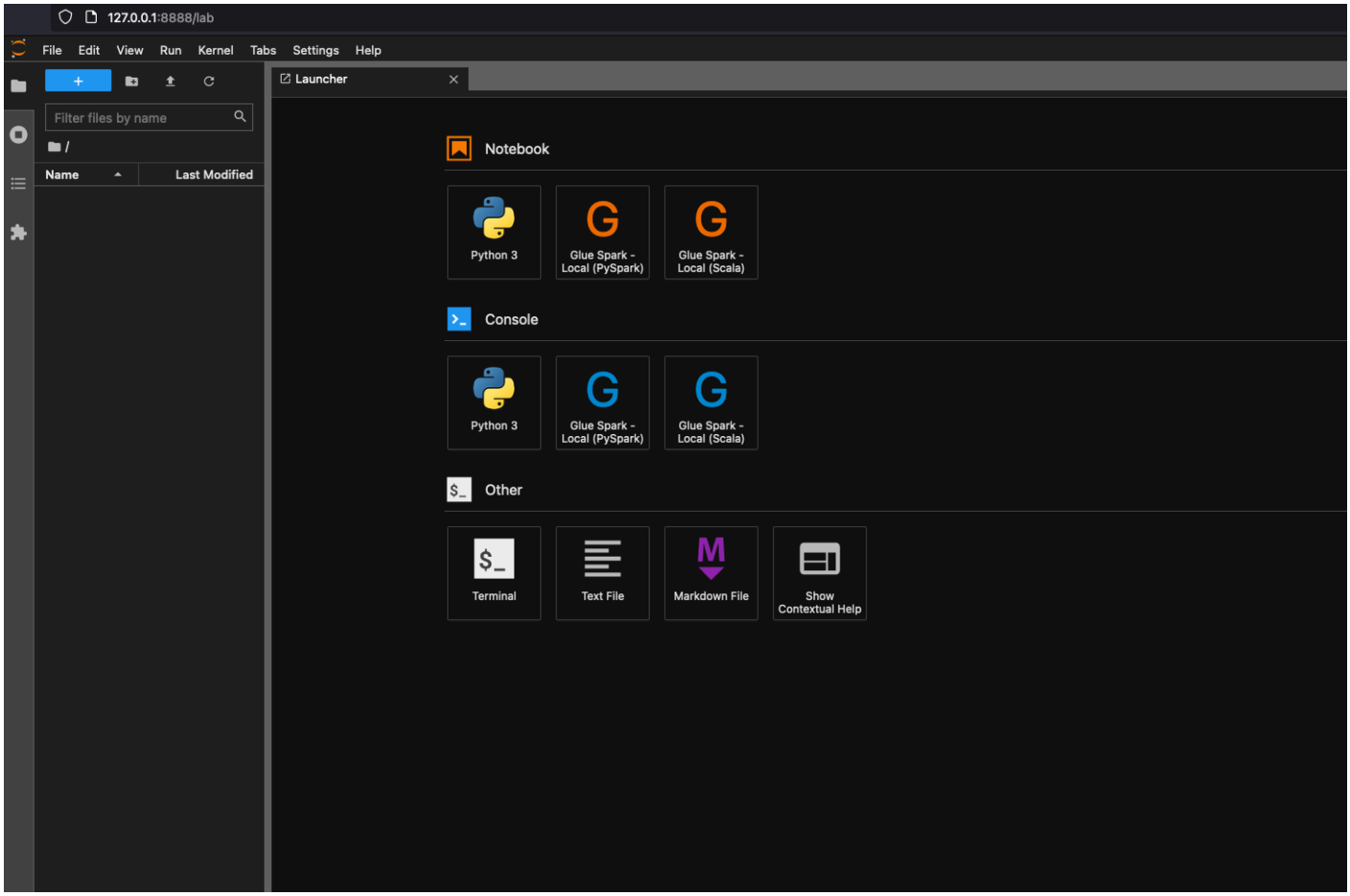
Laboratorio de Jupyter

Puede iniciar Jupyter para desarrollo interactivo y consultas ad hoc en los cuadernos.

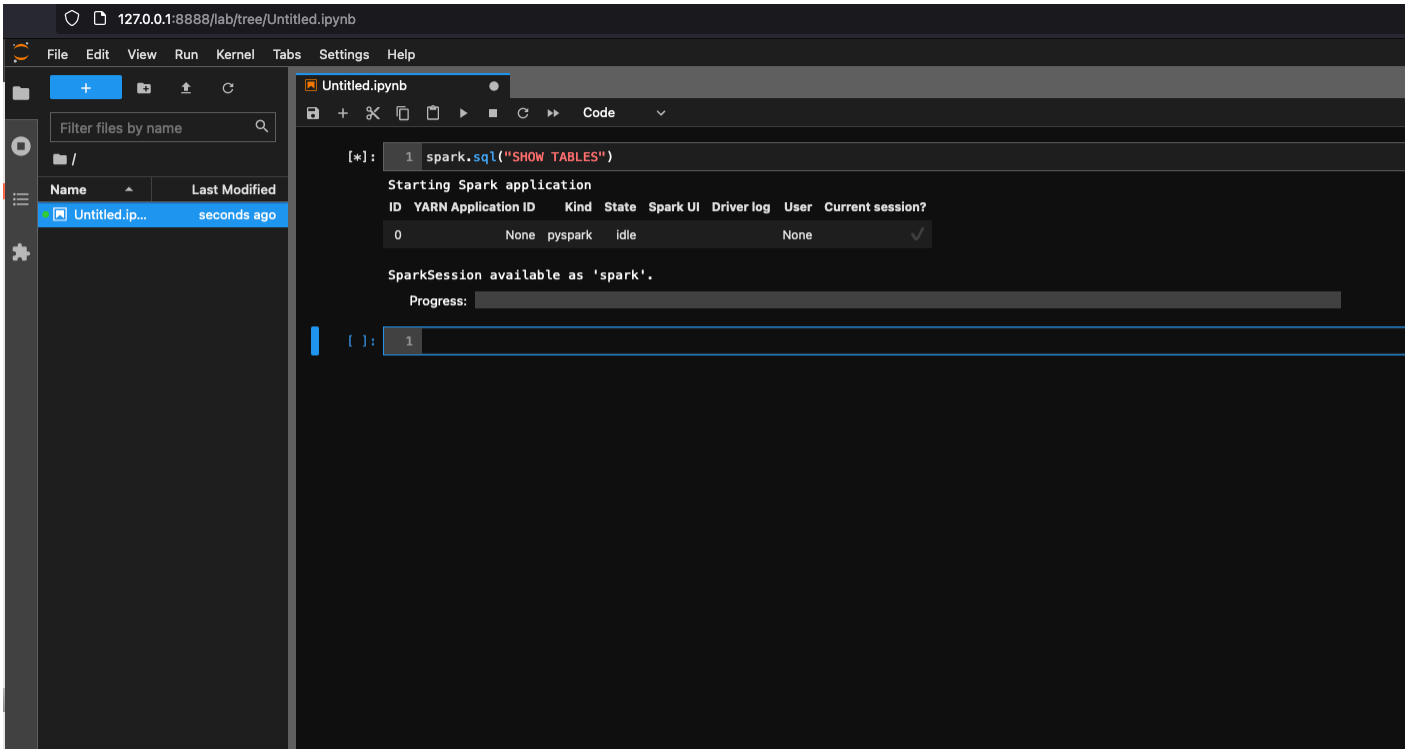
1. Ejecute el siguiente comando para iniciar Jupyter Lab:

```
$ JUPYTER_WORKSPACE_LOCATION=/local_path_to_workspace/jupyter_workspace/
$ docker run -it -v ~/.aws:/home/glue_user/.aws -v $JUPYTER_WORKSPACE_LOCATION:/
home/glue_user/workspace/jupyter_workspace/ -e AWS_PROFILE=$PROFILE_NAME -e
 DISABLE_SSL=true --rm -p 4040:4040 -p 18080:18080 -p 8998:8998 -p 8888:8888 --name
 glue_jupyter_lab amazon/aws-glue-libs:glue_libs_4.0.0_image_01 /home/glue_user/
jupyter/jupyter_start.sh
...
[I 2022-01-24 08:19:21.368 ServerApp] Serving notebooks from local directory: /home/
glue_user/workspace/jupyter_workspace
[I 2022-01-24 08:19:21.368 ServerApp] Jupyter Server 1.13.1 is running at:
[I 2022-01-24 08:19:21.368 ServerApp] http://faa541f8f99f:8888/lab
[I 2022-01-24 08:19:21.368 ServerApp] or http://127.0.0.1:8888/lab
[I 2022-01-24 08:19:21.368 ServerApp] Use Control-C to stop this server and shut down
 all kernels (twice to skip confirmation).
```

2. Abra http://127.0.0.1:8888/lab en el navegador web de su máquina local para ver la interfaz de usuario de Jupyter lab.



3. Elija Glue Spark Local (PySpark) en Notebook (Cuaderno). Puede comenzar a desarrollar el código en la interfaz de usuario interactiva del cuaderno de Jupyter.



Configuración del contenedor para utilizar Visual Studio Code

Requisitos previos:

1. Instale Visual Studio Code.
2. Instalación de Python.
3. Instalar Visual Studio Code Remote: contenedores
4. Abra la carpeta de WorkSpace en Visual Studio Code.
5. Elija Configuración.
6. Elija WorkSpace.
7. Elija Abrir Settings (JSON) (Abrir configuración).
8. Pegue el siguiente JSON y guárdelo.

```
{
    "python.defaultInterpreterPath": "/usr/bin/python3",
    "python.analysis.extraPaths": [
        "/home/glue_user/aws-glue-libs/PyGlue.zip:/home/glue_user/spark/python/lib/
py4j-0.10.9-src.zip:/home/glue_user/spark/python/",
    ]
```

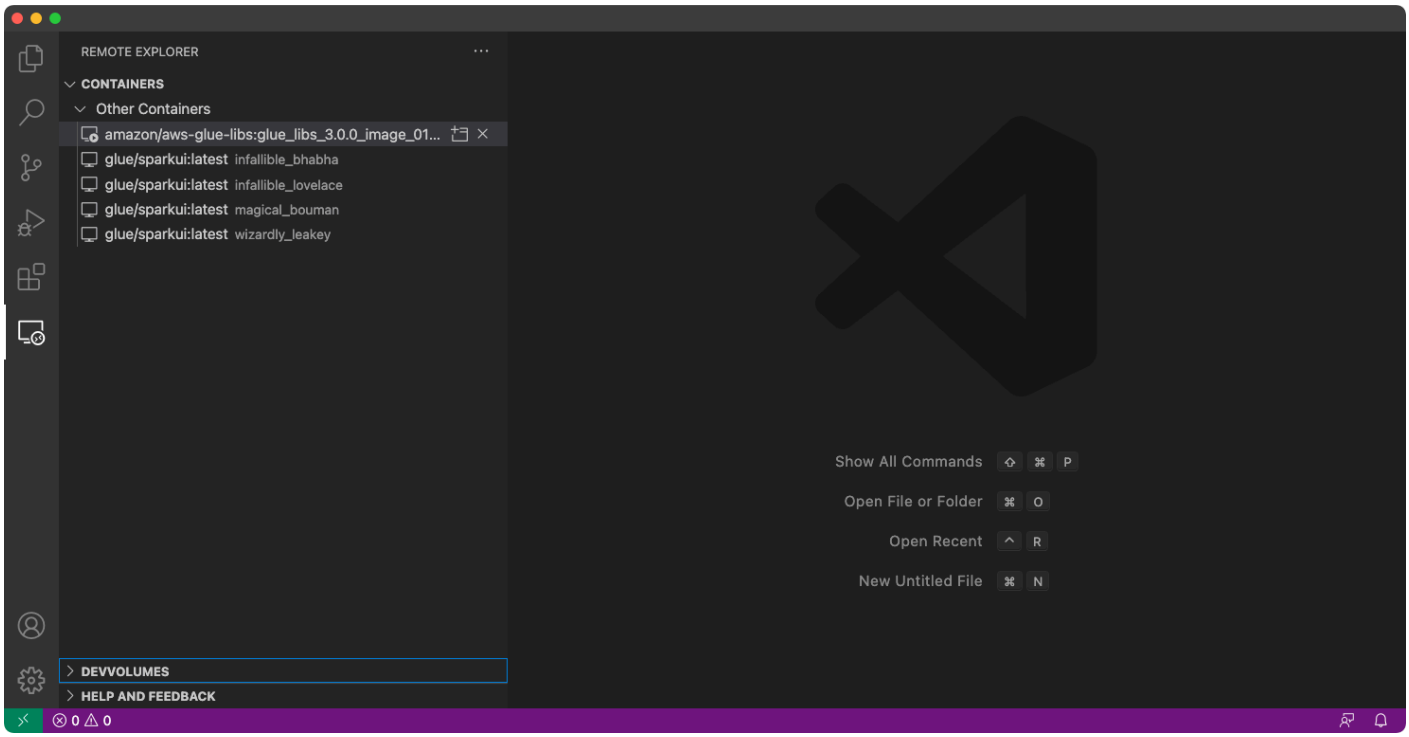
```
}
```

Pasos:

1. Ejecute el contenedor Docker.

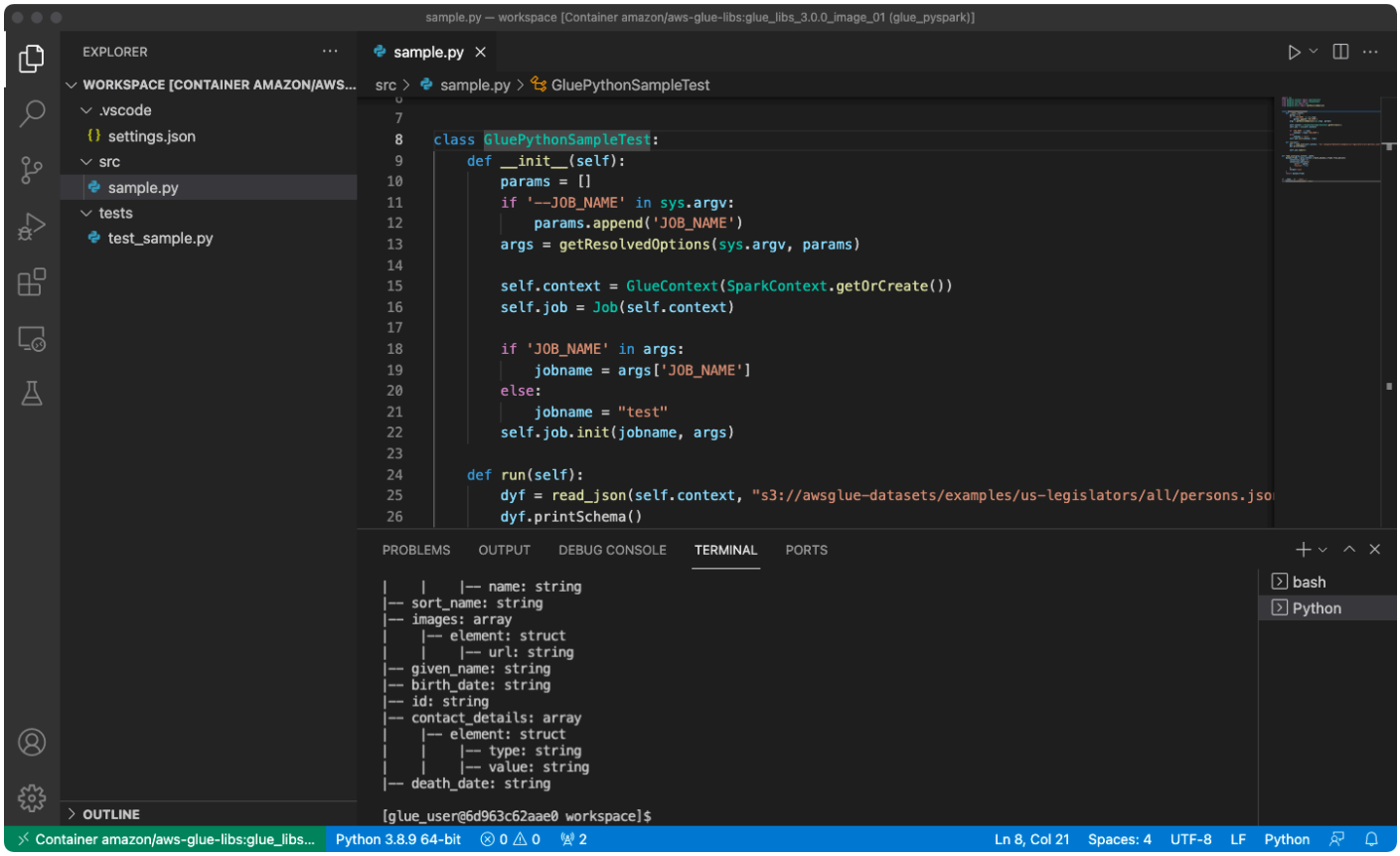
```
$ docker run -it -v ~/.aws:/home/glue_user/.aws -v $WORKSPACE_LOCATION:/
home/glue_user/workspace/ -e AWS_PROFILE=$PROFILE_NAME -e DISABLE_SSL=true
 --rm -p 4040:4040 -p 18080:18080 --name glue_pyspark amazon/aws-glue-
libs:glue_libs_4.0.0_image_01 pyspark
```

2. Inicie Visual Studio Code.
3. Elija Remote Explorer (Explorador remoto) en el menú de la izquierda y elija `amazon/aws-glue-libs:glue_libs_4.0.0_image_01`.



4. Haga clic con el botón derecho y elija Attach to Container (Adjuntar al contenedor). Si aparece un cuadro de diálogo, elija Got it (Entendido).
5. Abra `/home/glue_user/workspace/`.
6. Cree un script de Glue PySpark y elija Run (Ejecutar).

   Verá la ejecución correcta del script.



Apéndice: código de muestra de trabajo de AWS Glue para pruebas

Este apéndice proporciona scripts como código de muestra de trabajo de AWS Glue para fines de prueba.

sample.py: código de muestra para utilizar la biblioteca de ETL de AWS Glue con una llamada a la API de Amazon S3

```
import sys
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job
from awsglue.utils import getResolvedOptions


class GluePythonSampleTest:
    def __init__(self):
        params = []
        if '--JOB_NAME' in sys.argv:
            params.append('JOB_NAME')
```

```
        args = getResolvedOptions(sys.argv, params)

        self.context = GlueContext(SparkContext.getOrCreate())
        self.job = Job(self.context)

        if 'JOB_NAME' in args:
            jobname = args['JOB_NAME']
        else:
            jobname = "test"
        self.job.init(jobname, args)

    def run(self):
        dyf = read_json(self.context, "s3://awsglue-datasets/examples/us-legislators/
all/persons.json")
        dyf.printSchema()

        self.job.commit()


def read_json(glue_context, path):
    dynamicframe = glue_context.create_dynamic_frame.from_options(
        connection_type='s3',
        connection_options={
            'paths': [path],
            'recurse': True
        },
        format='json'
    )
    return dynamicframe


if __name__ == '__main__':
    GluePythonSampleTest().run()
```

El código anterior requiere permisos de Amazon S3 en IAM de AWS. Debe conceder la política administrada de IAM de `arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess` o una política personalizada de IAM que le permite llamar a `ListBucket` y `GetObject` para la ruta de Amazon S3.

test_sample.py: Código de muestra para la prueba de unidad de sample.py.

```
import pytest
from pyspark.context import SparkContext
```

```
from awsglue.context import GlueContext
from awsglue.job import Job
from awsglue.utils import getResolvedOptions
import sys
from src import sample


@pytest.fixture(scope="module", autouse=True)
def glue_context():
    sys.argv.append('--JOB_NAME')
    sys.argv.append('test_count')

    args = getResolvedOptions(sys.argv, ['JOB_NAME'])
    context = GlueContext(SparkContext.getOrCreate())
    job = Job(context)
    job.init(args['JOB_NAME'], args)

    yield(context)

    job.commit()


def test_counts(glue_context):
    dyf = sample.read_json(glue_context, "s3://awsglue-datasets/examples/us-
legislators/all/persons.json")
    assert dyf.toDF().count() == 1961
```

Desarrollo mediante la biblioteca de ETL de AWS Glue

La biblioteca ETL de AWS Glue se encuentra disponible en un bucket público de Amazon S3, y puede utilizarla el sistema de compilación Apache Maven. Esto le permite desarrollar y probar localmente sus scripts de extracción, transformación y carga (ETL) de Python y Scala, sin necesidad de una conexión de red. Se recomienda el desarrollo local con la imagen de Docker, ya que proporciona un entorno correctamente configurado para el uso de esta biblioteca.

El desarrollo local se encuentra disponible para todas las versiones de AWS Glue, incluso las versiones 0.9, 1.0, 2.0 y posteriores de AWS Glue. Para obtener información acerca de las versiones de Python y Apache Spark que están disponibles con AWS Glue, consulte la Glue version job property.

La biblioteca es accesible entrega con la licencia de software de Amazon (https://aws.amazon.com/asl).

Restricciones de desarrollo local

Tenga en cuenta las siguientes restricciones al utilizar la biblioteca Scala de AWS Glue para desarrollar localmente.

- Evite crear un jar de ensamblado ("fat jar" o "uber jar") con la biblioteca de AWS Glue, porque esto hará que se deshabiliten las siguientes características:
  - Marcadores de trabajo
  - Escritor de AWS Glue Parquet (Uso del formato Parquet en AWS Glue)
  - Transformación FillMissingValues (Scala o Python)

  Estas características solo están disponibles en el sistema de trabajos de AWS Glue.
- No se admite la transformación FindMatches en el desarrollo local.
- El lector vectorizado SIMD CSV no es compatible con el desarrollo local.
- El desarrollo local no admite la propiedad customJdbcDriverS3Path para cargar el controlador JDBC desde una ruta S3. Como alternativa, puede descargar el controlador JDBC en su local y cargarlo desde allí.
- La Calidad de datos de Glue no se admite en el desarrollo local.

Desarrollo a nivel local con Python

Complete algunos pasos de requisitos previos y, a continuación, use las utilidades de AWS Glue para probar y enviar el script de ETL de Python.

Requisitos previos para el desarrollo local de Python

Complete estos pasos para prepararse para el desarrollo local de Python:

1. Clonar el repositorio de AWS Glue Python desde GitHub (https://github.com/awslabs/aws-glue-libs).
2. Realice una de las acciones siguientes:
   - Para la versión 0.9 de AWS Glue, consulte la ramificación `glue-0.9`.
   - Para las versiones 1.0 de AWS Glue, consulte la ramificación `glue-1.0`. Todas las versiones de AWS Glue posteriores a 0.9 admiten Python 3.
   - Para las versiones 2.0 de AWS Glue, consulte la ramificación `glue-2.0`.
   - Para las versiones 3.0 de AWS Glue, consulte la ramificación `glue-3.0`.

- Para la versión 4.0 de AWS Glue, consulte la ramificación `master`.

3. Instale Apache Maven desde la siguiente ubicación: https://aws-glue-etl-artifacts.s3.amazonaws.com/glue-common/apache-maven-3.6.0-bin.tar.gz.
4. Instale la distribución de Apache Spark desde una de las siguientes ubicaciones:
   - Para la versión 0.9 de AWS Glue: https://aws-glue-etl-artifacts.s3.amazonaws.com/glue-0.9/spark-2.2.1-bin-hadoop2.7.tgz
   - Para la versión 1.0 de AWS Glue: https://aws-glue-etl-artifacts.s3.amazonaws.com/glue-1.0/spark-2.4.3-bin-hadoop2.8.tgz
   - Para la versión 2.0 de AWS Glue: https://aws-glue-etl-artifacts.s3.amazonaws.com/glue-2.0/spark-2.4.3-bin-hadoop2.8.tgz
   - Para la versión 3.0 de AWS Glue: https://aws-glue-etl-artifacts.s3.amazonaws.com/glue-3.0/spark-3.1.1-amzn-0-bin-3.2.1-amzn-3.tgz
   - Para la versión 4.0 AWS Glue: https://aws-glue-etl-artifacts.s3.amazonaws.com/glue-4.0/spark-3.3.0-amzn-1-bin-3.3.3-amzn-0.tgz
5. Exporte la variable de entorno `SPARK_HOME`, estableciéndola en la ubicación raíz extraída del archivo Spark. Por ejemplo:
   - Para la versión 0.9 de AWS Glue: `export SPARK_HOME=/home/$USER/spark-2.2.1-bin-hadoop2.7`
   - Para las versiones 1.0 y 2.0 de AWS Glue: `export SPARK_HOME=/home/$USER/spark-2.4.3-bin-spark-2.4.3-bin-hadoop2.8`
   - Para la versión 3.0 de AWS Glue: `export SPARK_HOME=/home/$USER/spark-3.1.1-amzn-0-bin-3.2.1-amzn-3`
   - Para la versión 4.0 AWS Glue: `export SPARK_HOME=/home/$USER/spark-3.3.0-amzn-1-bin-3.3.3-amzn-0`

Ejecución del script de ETL de Python

Con los archivos jar de AWS Glue disponibles para desarrollo local, puede ejecutar localmente el paquete AWS Glue Python.

Utilice las siguientes utilidades y marcos de trabajo para probar y ejecutar el script de Python. Los comandos enumerados en la siguiente tabla se ejecutan desde el directorio raíz del paquete AWS Glue Python.

Utilidad	Comando	Descripción
Intérprete de comandos de AWS Glue	`./bin/gluepyspark`	Escribe y ejecuta scripts de Python en un intérprete de comandos que se integra con bibliotecas de ETL de AWS Glue.
AWS Glue Submit (Enviar).	`./bin/gluesparksubmit`	Envía un script de Python completo para su ejecución.
Pytest	`./bin/gluepytest`	Escribe y ejecuta pruebas unitarias de su código Python. El módulo pytest debe estar instalado y disponible en `PATH`. Para obtener más información, consulte la documentación de pytest.

Desarrollo a nivel local con Scala

Complete algunos pasos de requisitos de previos y, a continuación, emita un comando Maven para ejecutar localmente el script de ETL de Scala.

Requisitos previos para el desarrollo local de Scala

Complete estos pasos a fin de prepararse para el desarrollo local de Scala.

Paso 1: Instalar el software

En este paso, instalará el software y establecerá la variable de entorno necesaria.

1. Instale Apache Maven desde la siguiente ubicación: https://aws-glue-etl-artifacts.s3.amazonaws.com/glue-common/apache-maven-3.6.0-bin.tar.gz.
2. Instale la distribución de Apache Spark desde una de las siguientes ubicaciones:
   - Para la versión 0.9 de AWS Glue: https://aws-glue-etl-artifacts.s3.amazonaws.com/glue-0.9/spark-2.2.1-bin-hadoop2.7.tgz
   - Para la versión 1.0 de AWS Glue: https://aws-glue-etl-artifacts.s3.amazonaws.com/glue-1.0/spark-2.4.3-bin-hadoop2.8.tgz
   - Para la versión 2.0 de AWS Glue: https://aws-glue-etl-artifacts.s3.amazonaws.com/glue-2.0/spark-2.4.3-bin-hadoop2.8.tgz

- Para la versión 3.0 de AWS Glue: https://aws-glue-etl-artifacts.s3.amazonaws.com/glue-3.0/spark-3.1.1-amzn-0-bin-3.2.1-amzn-3.tgz
- Para la versión 4.0 AWS Glue: https://aws-glue-etl-artifacts.s3.amazonaws.com/glue-4.0/spark-3.3.0-amzn-1-bin-3.3.3-amzn-0.tgz

3. Exporte la variable de entorno `SPARK_HOME`, estableciéndola en la ubicación raíz extraída del archivo Spark. Por ejemplo:
   - Para la versión 0.9 de AWS Glue: `export SPARK_HOME=/home/$USER/spark-2.2.1-bin-hadoop2.7`
   - Para las versiones 1.0 y 2.0 de AWS Glue: `export SPARK_HOME=/home/$USER/spark-2.4.3-bin-spark-2.4.3-bin-hadoop2.8`
   - Para la versión 3.0 de AWS Glue: `export SPARK_HOME=/home/$USER/spark-3.1.1-amzn-0-bin-3.2.1-amzn-3`
   - Para la versión 4.0 AWS Glue: `export SPARK_HOME=/home/$USER/spark-3.3.0-amzn-1-bin-3.3.3-amzn-0`

Paso 2: Configurar su proyecto de Maven

Utilice el siguiente archivo `pom.xml` como plantilla para sus aplicaciones de Scala de AWS Glue. Contiene los elementos `dependencies`, `repositories` y `plugins` necesarios. Reemplace la cadena de `Glue version` por uno de los siguientes valores:

- `4.0.0` para la versión 4.0 de AWS Glue
- Para la versión 3.0 de AWS Glue: `3.0.0`
- Para las versiones 1.0 y 2.0 de AWS Glue: `1.0.0`
- Para la versión 0.9 de AWS Glue: `0.9.0`

```
  <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://
maven.apache.org/maven-v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.amazonaws</groupId>
    <artifactId>AWSGlueApp</artifactId>
    <version>1.0-SNAPSHOT</version>
    <name>${project.artifactId}</name>
    <description>AWS ETL application</description>
```

```
        <properties>
            <scala.version>2.11.1 for AWS Glue 2.0 or below, 2.12.7 for AWS Glue 3.0
 and 4.0</scala.version>
            <glue.version>Glue version with three numbers (as mentioned earlier)</
glue.version>
        </properties>
    <dependencies>
        <dependency>
            <groupId>org.scala-lang</groupId>
            <artifactId>scala-library</artifactId>
            <version>${scala.version}</version>
   <!-- A "provided" dependency, this will be ignored when you package your application
 -->
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>com.amazonaws</groupId>
            <artifactId>AWSGlueETL</artifactId>
   <version>${glue.version}</version>
            <!-- A "provided" dependency, this will be ignored when you package your
 application -->
            <scope>provided</scope>
        </dependency>
    </dependencies>

    <repositories>
        <repository>
            <id>aws-glue-etl-artifacts</id>
            <url>https://aws-glue-etl-artifacts.s3.amazonaws.com/release/</url>
        </repository>
    </repositories>
    <build>
        <sourceDirectory>src/main/scala</sourceDirectory>
        <plugins>
            <plugin>
                <!-- see http://davidb.github.com/scala-maven-plugin -->
                <groupId>net.alchim31.maven</groupId>
                <artifactId>scala-maven-plugin</artifactId>
                <version>3.4.0</version>
                <executions>
                    <execution>
                        <goals>
                            <goal>compile</goal>
                            <goal>testCompile</goal>
```

```
                    </goals>
                </execution>
            </executions>
        </plugin>
        <plugin>
            <groupId>org.codehaus.mojo</groupId>
            <artifactId>exec-maven-plugin</artifactId>
            <version>1.6.0</version>
            <executions>
                <execution>
                    <goals>
                    <goal>java</goal>
                    </goals>
                </execution>
            </executions>
            <configuration>
            <systemProperties>
                <systemProperty>
                    <key>spark.master</key>
                    <value>local[*]</value>
                </systemProperty>
                <systemProperty>
                    <key>spark.app.name</key>
                    <value>localrun</value>
                </systemProperty>
                <systemProperty>
                    <key>org.xerial.snappy.lib.name</key>
                    <value>libsnappyjava.jnilib</value>
                </systemProperty>
            </systemProperties>
            </configuration>
        </plugin>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-enforcer-plugin</artifactId>
            <version>3.0.0-M2</version>
            <executions>
                <execution>
                    <id>enforce-maven</id>
                    <goals>
                        <goal>enforce</goal>
                    </goals>
                    <configuration>
                        <rules>
```

```
                                <requireMavenVersion>
                                    <version>3.5.3</version>
                                </requireMavenVersion>
                            </rules>
                        </configuration>
                    </execution>
                </executions>
            </plugin>
  <!-- The shade plugin will be helpful in building a uberjar or fatjar.
  You can use this jar in the AWS Glue runtime environment. For more information, see
 https://maven.apache.org/plugins/maven-shade-plugin/ -->
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-shade-plugin</artifactId>
                <version>3.2.4</version>
                <configuration>
                    <!-- any other shade configurations -->
                </configuration>
                <executions>
                    <execution>
                        <phase>package</phase>
                        <goals>
                            <goal>shade</goal>
                        </goals>
                    </execution>
                </executions>
            </plugin>
        </plugins>
    </build>
</project>
```

Ejecución del script de ETL de Scala

Ejecute el siguiente comando desde el directorio raíz del proyecto Maven para ejecutar el script de ETL de Scala.

```
mvn exec:java -Dexec.mainClass="mainClass" -Dexec.args="--JOB-NAME jobName"
```

Sustituya *mainClass* por el nombre de clase completo de la clase principal del script. Sustituya *jobName* por el nombre de trabajo que desee.

Configuración de un entorno de pruebas

Para obtener ejemplos acerca de cómo configurar un entorno local de pruebas, consulte los siguientes artículos de blog:

- Building an AWS Glue ETL pipeline locally without an AWS account (Creación local de una canalización de ETL de AWS Glue sin una cuenta de AWS)
- Developing AWS Glue ETL jobs locally using a container (Desarrollo local de trabajos de ETL de AWS Glue con un contenedor)

Si desea utilizar puntos de enlace de desarrollo o cuaderno para probar sus scripts ETL, consulte Desarrollo de scripts usando puntos de conexión de desarrollo.

> **Note**
>
> Los trabajos de la versión 2.0 de AWS Glue no soportan el uso de puntos de enlace desarrollo. Para obtener más información, consulte Ejecución de trabajos de ETL de Spark con tiempos de inicio reducidos.

Puntos de conexión de desarrollo

> **Note**
>
> La experiencia de consola para los puntos de conexión de desarrollo se eliminó a partir del 31 de marzo de 2023. La creación, actualización y supervisión de los puntos de conexión de desarrollo siguen estando disponibles a través de la API de puntos de conexión de desarrollo y de la CLI de AWS Glue.

Se recomienda migrar de los puntos de conexión de desarrollo a las sesiones interactivas por los motivos que se indican a continuación. Para conocer las acciones necesarias sobre cómo migrar de puntos de conexión de desarrollo a sesiones interactivas, consulte Migración de puntos de conexión de desarrollo a sesiones interactivas.

Descripción	Puntos de conexión de desarrollo	Sesiones interactivas
Compatibilidad de la versión de Glue	Compatible con las versiones 0.9 y 1.0 de AWS Glue	Compatible con las versiones 2.0 y posteriores de AWS Glue
Los puntos de conexión de desarrollo no están disponibles en Asia-Pacífico (Yakarta) (`ap-southeast-3`), Medio Oriente (UAE) (`me-central-1`), Europa (España) (`eu-south-2`), Europa (Zúrich) (`eu-central-2`) ni en otras regiones nuevas que se establezcan en el futuro	Las sesiones interactivas no están disponibles actualmente en la región Medio Oriente (EAU) (`me-central-1`), pero es posible que estén disponibles más adelante.	
Método de acceso al clúster de Spark	Compatible con SSH, intérprete de comandos de REPL, cuaderno de Jupyter, IDE (por ejemplo, PyCharm)	Compatible con cuaderno de AWS Glue Studio, cuaderno de Jupyter, varios IDE (por ejemplo, Visual Studio Code, PyCharm) y cuaderno de SageMaker
Tiempo de la primera consulta	Se necesitan entre 10 y 15 minutos para configurar un clúster de Spark	Puede tardar hasta 1 minuto en configurar un clúster de Spark efímero
Modelo de precios	AWS cobra por los puntos de conexión de desarrollo en función del momento en que se aprovisiona el punto de conexión y la cantidad de DPU. El tiempo de espera de los puntos de conexión de desarrollo no se agota.	AWS cobra por las sesiones interactivas en función del tiempo que la sesión está activa y de la cantidad de unidades de procesamiento de datos (DPU). Las sesiones interactivas tienen tiempos de inactividad configurables.

Descripción	Puntos de conexión de desarrollo	Sesiones interactivas
	Hay una duración mínima de facturación de 10 minutos para cada punto de conexión de desarrollo aprovisionado. Además, AWS cobra por el cuaderno de Jupyter en instancias de Amazon EC2 y los cuadernos de SageMaker cuando los configura con puntos de conexión de desarrollo.	Los cuadernos de AWS Glue Studio proporcionan una interfaz integrada para las sesiones interactivas y se ofrecen sin costo adicional . Hay una duración mínima de facturación de 1 minuto para cada sesión interactiva. Los cuadernos de AWS Glue Studio ofrecen una interfaz integrada para sesiones interactivas y se ofrecen sin costo adicional
Experiencia de consola	Solo está disponible a través de la CLI y la API	Disponible a través de la consola de AWS Glue, la CLI y las API

Migración de puntos de conexión de desarrollo a sesiones interactivas

Utilice la siguiente lista de comprobación para determinar el método adecuado para migrar de los puntos de conexión de desarrollo a las sesiones interactivas.

¿Su script depende de características específicas de la versión 0.9 o 1.0 de AWS Glue (por ejemplo, HDFS, YARN, etc.)?

Si la respuesta es sí, consulte Migrar trabajos de AWS Glue a la versión 3.0 de AWS Glue para obtener información sobre cómo migrar de Glue 0.9 o 1.0 a Glue 3.0 y versiones posteriores.

¿Qué método utiliza para acceder al punto de conexión de desarrollo?

Si utiliza este método	Haga esto
Cuaderno de SageMaker, cuaderno de Jupyter o laboratorio de Jupyter	Para migrar al cuaderno de AWS Glue Studio, descargue los archivos `.ipynb` en Jupyter

Si utiliza este método	Haga esto
	y cree un nuevo trabajo de cuaderno de AWS Glue Studio mediante la carga del archivo `.ipynb`. También puede utilizar SageMaker Studio y seleccionar el kernel de AWS Glue.
Bloc de notas de Zeppelin	Para convertir el cuaderno en un cuaderno de Jupyter de forma manual, copie y pegue el código; para hacerlo de forma automática, utilice un conversor de terceros, como ze2nb. A continuación, utilice el cuaderno en el cuaderno de AWS Glue Studio o en SageMaker Studio.
IDE	Consulte Author AWS Glue jobs with PyCharm using AWS Glue interactive sessions o Uso de las sesiones interactivas con Microsoft Visual Studio Code.
REPL	Para instalar `aws-glue-session package` de manera local, ejecute el siguiente comando: • Para Python: `jupyter console --kernal glue_pyspark` • Para Scala: `jupyter console --kernal glue_spark`
SSH	No hay ninguna opción correspondiente en las sesiones interactivas. También puede utilizar una imagen de Docker. Para obtener más información, consulte Desarrollo mediante una imagen de Docker.

En las siguientes secciones, se ofrece información sobre el uso de puntos de conexión de desarrollo para desarrollar trabajos en AWS Glue versión 1.0.

Temas

- Desarrollo de scripts usando puntos de conexión de desarrollo
- Administración de blocs de notas

Desarrollo de scripts usando puntos de conexión de desarrollo

> **Note**
>
> Los puntos de conexión de desarrollo solo son compatibles con versiones de AWS Glue anteriores a la 2.0. Si busca un entorno interactivo en el que pueda crear y probar scripts de ETL, utilice los Blocs de notas en AWS Glue Studio.

AWS Glue puede crear un entorno, conocido como punto de enlace de desarrollo, que puede utilizar para desarrollar y probar de forma iterativa los scripts de extracción, transformación y carga (ETL). Puede crear, editar y eliminar puntos de enlace de desarrollo mediante la consola o la API de AWS Glue.

Administración de su entorno de desarrollo

Al crear un punto de enlace de desarrollo, debe proporcionar valores de configuración para aprovisionar el entorno de desarrollo. Estos valores indican a AWS Glue cómo configurar la red para que pueda obtener acceso al punto de enlace de forma segura y que el punto de enlace pueda obtener acceso a sus almacenes de datos.

A continuación puede crear un bloc de notas que se conecte al punto de enlace y usar su bloc de notas para crear y probar su script ETL. Cuando esté satisfecho con los resultados de su proceso de desarrollo, podrá crear un flujo de trabajo de ETL que ejecute su script. Con este proceso, puede añadir funciones y depurar sus scripts de forma interactiva.

Siga los tutoriales de esta sección para aprender a utilizar su punto de enlace de desarrollo con blocs de notas.

Temas

- Flujo de trabajo de punto de conexión de desarrollo
- Cómo trabajan los puntos de conexión de desarrollo de AWS Glue con cuadernos de SageMaker
- Añadir un punto de conexión de desarrollo.
- Acceso al punto de conexión de desarrollo

- Tutorial: configuración de un cuaderno de Jupyter en JupyterLab para probar y depurar scripts ETL
- Tutorial: uso de un bloc de notas de SageMaker con su punto de conexión de desarrollo
- Tutorial: Utilice un REPL Shell con su punto de conexión de desarrollo
- Tutorial: Configuración de PyCharm Professional con un punto de conexión de desarrollo
- Configuración avanzada: uso compartido de puntos de conexión de desarrollo entre múltiples usuarios

Flujo de trabajo de punto de conexión de desarrollo

Para usar un punto de enlace de desarrollo de AWS Glue, puede seguir este flujo de flujo de trabajo.

1. Cree un punto de conexión de desarrollo mediante la API. El punto de enlace se lanza en una nube virtual privada (VPC) con sus grupos de seguridad definidos.
2. La API sondea el punto de conexión de desarrollo hasta que se aprovisione y esté listo para funcionar. Cuando esté listo, conéctese al punto de enlace de desarrollo utilizando uno de los siguientes métodos para crear y probar scripts AWS Glue.
   - Cree un bloc de notas de SageMaker en su cuenta. Para obtener más información acerca de cómo crear un blocs de notas, consulte the section called “Creación de código con cuadernos de AWS Glue Studio”.
   - Abra una ventana de terminal para conectarse directamente a un punto de enlace de desarrollo.
   - Si tiene la edición Professional del IDE de Python de PyCharm de JetBrains, conéctela a un punto de enlace de desarrollo y úsela para su desarrollo interactivo. Si inserta instrucciones `pydevd` en su script, PyCharm puede admitir puntos de interrupción remotos.
3. Cuando termine de depurar y realizar pruebas en su punto de enlace de desarrollo, puede eliminarlo.

Cómo trabajan los puntos de conexión de desarrollo de AWS Glue con cuadernos de SageMaker

Una de las formas comunes de acceder a sus puntos de enlace de desarrollo es usar Jupyter en los blocs de notas de SageMaker. El bloc de notas de Jupyter es una aplicación web de código abierto que se utiliza ampliamente en visualización, análisis, machine learning, etc. Un bloc de notas de SageMaker de AWS Glue le ofrece una experiencia de bloc de notas de Jupyter con puntos de

enlace de desarrollo de AWS Glue. En el bloc de notas de SageMaker de AWS Glue, el entorno del bloc de notas de Jupyter está preconfigurado con SparkMagic, un complemento de Jupyter de código abierto para enviar trabajos de Spark a un clúster remoto de Spark. Apache Livy es un servicio que permite la interacción con un clúster de Spark remoto a través de una API REST. En el bloc de notas de SageMaker de AWS Glue, SparkMagic está configurado para invocar la API REST hacia un servidor Livy que se ejecuta en un punto de enlace de desarrollo de AWS Glue.

El siguiente flujo de texto explica cómo funciona cada componente:

Bloc de notas de SageMaker de AWS Glue: (Jupyter → SparkMagic) → (red) →punto de conexión de desarrollo de AWS Glue: (Apache Livy → Apache Spark)

Una vez que ejecute el script de Spark escrito en cada párrafo en un bloc de notas de Jupyter, el código de Spark se envía al servidor Livy a través de SparkMagic, luego se ejecuta un trabajo de Spark llamado “livy-session-n” en el clúster de Spark. Este trabajo se llama sesión de Livy. El trabajo de Spark se ejecutará mientras la sesión del bloc de notas esté activa. El trabajo de Spark finalizará cuando cierre el kernel de Jupyter desde el bloc de notas, o cuando se agote el tiempo de espera de la sesión. Se inicia un trabajo de Spark por archivo de bloc de notas (.ipynb).

Puede utilizar un único punto de enlace de desarrollo de AWS Glue con varias instancias de bloc de notas de SageMaker. Puede crear varios archivos de bloc de notas en cada instancia de bloc de notas de SageMaker. Al abrir un archivo de cada bloc de notas y ejecutar los párrafos, se inicia una sesión de Livy por archivo de bloc de notas en el clúster de Spark a través de SparkMagic. Cada sesión de Livy corresponde a un solo trabajo de Spark.

Comportamiento predeterminado de los puntos de conexión de desarrollo de AWS Glue y los cuadernos de SageMaker

Los trabajos de Spark se ejecutan en función de la configuración de Spark. Hay varias formas de establecer la configuración de Spark (por ejemplo, configuración de clúster de Spark, configuración de SparkMagic, etc.).

De forma predeterminada, Spark asigna recursos del clúster a una sesión de Livy en función de la configuración del clúster de Spark. En los puntos de desarrollo de AWS Glue, la configuración del clúster depende del tipo de empleado. A continuación, se incluye una tabla que explica las configuraciones comunes por tipo de empleado.

	Estándar	G.1 X	G.2 X
`spark.driver.memory`	5 G	10 G	20 G
`spark.executor.memory`	5 G	10 G	20 G
`spark.executor.cores`	4	8	16
`spark.dynamicAllocation.enabled`	TRUE	TRUE	TRUE

El número máximo de ejecutores de Spark se calcula automáticamente mediante una combinación de DPU (o `NumberOfWorkers`) y tipo de empleado.

	Estándar	G.1 X	G.2 X
Cantidad máxima de ejecutores de Spark	`(DPU - 1) * 2 - 1`	`(NumberOfWorkers - 1)`	`(NumberOfWorkers - 1)`

Por ejemplo, si el punto de enlace de desarrollo tiene 10 empleados y el tipo de empleado es `G.1X`, entonces tendrá 9 ejecutores de Spark y todo el clúster tendrá 90 G de memoria del ejecutor ya que cada ejecutor tendrá 10 G de memoria.

Independientemente del tipo de empleado especificado, se activará la asignación dinámica de recursos de Spark. Si un conjunto de datos es lo suficientemente grande, Spark puede asignar todos los ejecutores a una única sesión de Livy ya que `spark.dynamicAllocation.maxExecutors`

no está configurado en forma predeterminada. Esto significa que otras sesiones de Livy en el mismo punto de enlace de desarrollo esperarán para lanzar nuevos ejecutores. Si el conjunto de datos es pequeño, Spark podrá asignar ejecutores a múltiples sesiones de Livy al mismo tiempo.

> **Note**
>
> Para obtener más información acerca de cómo se asignan los recursos en diferentes casos de uso y cómo se establece una configuración para modificar el comportamiento, consulte Configuración avanzada: uso compartido de puntos de conexión de desarrollo entre múltiples usuarios.

Añadir un punto de conexión de desarrollo.

Utilice puntos de enlace de desarrollo para desarrollar y probar iterativamente los scripts de extracción, transformación y carga (ETL) en AWS Glue. El trabajo con puntos de conexión de desarrollo solo está disponible a través de AWS Command Line Interface.

1. En una ventana de línea de comandos, escriba un comando similar al siguiente.

   ```
   aws glue create-dev-endpoint --endpoint-name "endpoint1" --role-arn
    "arn:aws:iam::account-id:role/role-name" --number-of-nodes "3" --glue-version
    "1.0" --arguments '{"GLUE_PYTHON_VERSION": "3"}' --region "region-name"
   ```

   Este comando especifica la versión de AWS Glue 1.0. Dado que esta versión es compatible con Python 2 y Python 3, puede utilizar el parámetro `arguments` para indicar la versión de Python deseada. Si se omite el parámetro `glue-version`, se presupone la versión de AWS Glue 0.9. Para obtener más información acerca de las versiones de AWS Glue, consulte Glue version job property.

   Para obtener información sobre parámetros de línea de comandos adicionales, consulte create-dev-endpoint en la Referencia de comandos de AWS CLI.

2. (Opcional) Escriba el siguiente comando para comprobar el estado del punto de enlace de desarrollo. Cuando el estado cambia a `READY`, el punto de enlace de desarrollo está listo para su uso.

   ```
   aws glue get-dev-endpoint --endpoint-name "endpoint1"
   ```

Acceso al punto de conexión de desarrollo

Al crear un punto de enlace de desarrollo en una nube virtual privada (VPC), AWS Glue devuelve solo una dirección IP privada. El campo de dirección IP pública no se rellena. Al crear un punto de enlace de desarrollo que no pertenece a la VPC, AWS Glue devuelve solamente una dirección IP pública.

Si el punto de enlace de desarrollo tiene una dirección pública, confirme que es accesible con la clave privada SSH para el punto de enlace de desarrollo, como en el siguiente ejemplo.

```
ssh -i dev-endpoint-private-key.pem glue@public-address
```

Supongamos que el punto de enlace de desarrollo tiene una dirección privada, la subred de la VPC es direccionable desde la red pública de Internet y sus grupos de seguridad permiten el acceso de entrada desde el cliente. En este caso, siga estos pasos para adjuntar una dirección IP elástica a un punto de enlace de desarrollo para permitir el acceso desde Internet.

> **Note**
>
> Si desea utilizar direcciones IP elásticas, la subred que se está utilizando requiere una gateway de Internet asociada a través de la tabla de ruteo.

Obtener acceso a un punto de enlace de desarrollo adjuntando una dirección IP elástica

1. Abra la consola de AWS Glue en https://console.aws.amazon.com/glue/.
2. En el panel de navegación, elija Dev endpoints (Puntos de enlace de desarrollo), y vaya a la página de detalles del punto de enlace de desarrollo. Registre el valor de Private address (Dirección privada) para su uso en el siguiente paso.
3. Abra la consola de Amazon EC2 en https://console.aws.amazon.com/ec2/.
4. En el panel de navegación, en Network & Security (Red y seguridad), seleccione Network Interfaces (Interfaces de red).
5. Busque el valor de Private DNS (IPv4) (DNS privado, IPv4) que corresponde al valor de Private address (Dirección privada) en la página de detalles del punto de enlace de desarrollo de la consola de AWS Glue.

Es posible que tenga que modificar las columnas que desea mostrar en su consola de Amazon EC2. Anote el valor de Network interface ID (ENI) (ID de interfaz de red) correspondiente a esta dirección (por ejemplo, `eni-12345678`.

6. En la consola de Amazon EC2, en Network & Security (Red y seguridad), elija Elastic IPs (Direcciones IP elásticas).
7. Elija Allocate new address (Asignar nueva dirección) y, a continuación, elija Allocate (Asignar) para asignar una nueva dirección IP elástica.
8. En la página Elastic IPs (IP elásticas), elija la nueva IP elástica asignada recientemente. A continuación, elija Actions (Acciones) y Associate address (Asociar dirección).
9. En la página Associate address (Asociar dirección), haga lo siguiente:

   - En Tipo de recurso, elija Interfaz de red.
   - En el campo Network interface (Interfaz de red), escriba el ID de interfaz de red (ENI) de la dirección privada.
   - Seleccione Asociar.
10. Confirme que la dirección IP elástica recién asociada es accesible con la clave privada SSH asociada al punto de enlace de desarrollo, como en el siguiente ejemplo.

    ```
    ssh -i dev-endpoint-private-key.pem glue@elastic-ip
    ```

    Para ver más información acerca de cómo utilizar un host bastión para obtener acceso de SSH a la dirección privada del punto de enlace de desarrollo, consulte la publicación del blog de seguridad de AWS Securely Connect to Linux Instances Running in a Private Amazon VPC (Conexión segura a instancias de Linux que se ejecutan en una VPC privada de Amazon).

Tutorial: configuración de un cuaderno de Jupyter en JupyterLab para probar y depurar scripts ETL

En este tutorial, se conecta un bloc de notas Jupyter en JupyterLab que se ejecuta en su máquina local a un punto de enlace de desarrollo. Esto se hace para poder ejecutar, depurar y probar de forma interactiva los scripts de ETL (extracción, transformación y carga) de AWS Glue antes de implementarlos. Este tutorial utiliza enrutamiento de puertos Secure Shell (SSH) para conectar su máquina local a un punto de enlace de desarrollo de AWS Glue. Para obtener más información, consulte Port forwarding (Enrutamiento de puertos) en Wikipedia.

Paso 1: instalar JupyterLab y Sparkmagic

Puede instalar JupyterLab con `conda` o `pip`. `conda` es un sistema de código abierto para la administración de paquetes y de entorno que se ejecuta en Windows, macOS y Linux. `pip` es el instalador del paquete para Python.

Si realizará la instalación en macOS, debe tener instalado Xcode para poder instalar Sparkmagic.

1. Instale JupyterLab, Sparkmagic y las extensiones relacionadas.

```
$ conda install -c conda-forge jupyterlab
$ pip install sparkmagic
$ jupyter nbextension enable --py --sys-prefix widgetsnbextension
$ jupyter labextension install @jupyter-widgets/jupyterlab-manager
```

2. Compruebe el directorio `sparkmagic` desde `Location`.

```
$ pip show sparkmagic | grep Location
Location: /Users/username/.pyenv/versions/anaconda3-5.3.1/lib/python3.7/site-
packages
```

3. Cambie su directorio al devuelto para `Location`, e instale los kernels para Scala y PySpark.

```
$ cd /Users/username/.pyenv/versions/anaconda3-5.3.1/lib/python3.7/site-packages
$ jupyter-kernelspec install sparkmagic/kernels/sparkkernel
$ jupyter-kernelspec install sparkmagic/kernels/pysparkkernel
```

4. Descargue un archivo `config` de prueba.

```
$ curl -o ~/.sparkmagic/config.json https://raw.githubusercontent.com/jupyter-
incubator/sparkmagic/master/sparkmagic/example_config.json
```

   En este archivo de configuración, puede configurar los parámetros relacionados con Spark como `driverMemory` y `executorCores`.

Paso 2: inicie JupyterLab

Cuando inicia JupyterLab, su navegador web predeterminado se abre automáticamente y se muestra la URL `http://localhost:8888/lab/workspaces/{workspace_name}`.

```
$ jupyter lab
```

Paso 3: inicie el enrutamiento de puertos SSH para conectarse a su punto de conexión de desarrollo

A continuación, use el enrutamiento de puertos locales SSH para redireccionar un puerto local (aquí, 8998) al destino remoto definido por AWS Glue (`169.254.76.1:8998`).

1. Abra una ventana de terminal independiente que le proporcione acceso a SSH. En Microsoft Windows, puede usar el shell de BASH proporcionado por Git for Windows (Git para Windows) o instale Cygwin.
2. Ejecute el siguiente comando SSH, modificado de la siguiente manera:

   - Reemplace *`private-key-file-path`* por una ruta al archivo `.pem` que contiene la clave privada que se corresponde con la clave pública que usó para crear su punto de enlace de desarrollo.
   - Si redirecciona un puerto distinto de 8998, reemplace 8998 por el número de puerto que usa realmente de forma local. La dirección, `169.254.76.1:8998`, es el puerto remoto y no la modifica el usuario.
   - Reemplace *`dev-endpoint-public-dns`* por la dirección DNS pública de su punto de enlace de desarrollo. Para encontrar esta dirección, diríjase a su punto de enlace de desarrollo en la consola de AWS Glue, elija el nombre y copie la Public address (Dirección pública) que se enumera en la página Endpoint details (Detalles del punto de enlace).

   ```
   ssh -i private-key-file-path -NTL 8998:169.254.76.1:8998 glue@dev-endpoint-public-
   dns
   ```

   Lo más probable es que vea un mensaje de advertencia similar al siguiente:

   ```
   The authenticity of host 'ec2-xx-xxx-xxx-xx.us-west-2.compute.amazonaws.com
    (xx.xxx.xxx.xx)'
   can't be established.  ECDSA key fingerprint is SHA256:4e97875Brt+1wKzRko
   +JflSnp21X7aTP3BcFnHYLEts.
   Are you sure you want to continue connecting (yes/no)?
   ```

   Ingrese **`yes`** y deje la ventana de terminal abierta mientras usa JupyterLab.
3. Compruebe que el enrutamiento de puertos SSH funciona correctamente con el punto de enlace de desarrollo.

```
$ curl localhost:8998/sessions
{"from":0,"total":0,"sessions":[]}
```

Paso 4: ejecute un fragmento de script sencillo en un párrafo del bloc de notas

Ahora su bloc de notas en JupyterLab debería funcionar con su punto de enlace de desarrollo. Escriba el siguiente fragmento de script en el bloc de notas y ejecútelo.

1. Compruebe que Spark se está ejecutando correctamente. El siguiente comando indica a Spark que calcule el 1 y luego imprima el valor.

   ```
   spark.sql("select 1").show()
   ```

2. Compruebe si la integración AWS Glue Data Catalog está funcionando. En el siguiente comando se muestran las tablas del Catálogo de datos.

   ```
   spark.sql("show tables").show()
   ```

3. Compruebe que un fragmento de script simple que use las bibliotecas de AWS Glue funcione.

   El script a continuación utiliza los metadatos de la tabla `persons_json` en AWS Glue Data Catalog para crear un `DynamicFrame` a partir de sus datos de ejemplo. A continuación, imprime el recuento de elementos y el esquema de estos datos.

```
import sys
from pyspark.context import SparkContext
from awsglue.context import GlueContext

# Create a Glue context
glueContext = GlueContext(SparkContext.getOrCreate())

# Create a DynamicFrame using the 'persons_json' table
persons_DyF = glueContext.create_dynamic_frame.from_catalog(database="legislators",
 table_name="persons_json")

# Print out information about *this* data
print("Count:  ", persons_DyF.count())
persons_DyF.printSchema()
```

La salida del script es la siguiente.

```
Count:  1961
root
|-- family_name: string
|-- name: string
|-- links: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- url: string
|-- gender: string
|-- image: string
|-- identifiers: array
|    |-- element: struct
|    |    |-- scheme: string
|    |    |-- identifier: string
|-- other_names: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- name: string
|    |    |-- lang: string
|-- sort_name: string
|-- images: array
|    |-- element: struct
|    |    |-- url: string
|-- given_name: string
|-- birth_date: string
|-- id: string
|-- contact_details: array
|    |-- element: struct
|    |    |-- type: string
|    |    |-- value: string
|-- death_date: string
```

Solución de problemas

- Durante la instalación de JupyterLab, si su equipo está detrás de un proxy corporativo o firewall, es posible que encuentre errores HTTP y SSL debido a perfiles de seguridad personalizados administrados por departamentos de TI corporativos.

  El siguiente es un ejemplo de un error típico que se produce cuando `conda` no puede conectarse a sus propios repositorios:

```
CondaHTTPError: HTTP 000 CONNECTION FAILED for url <https://repo.anaconda.com/pkgs/
main/win-64/current_repodata.json>
```

Esto puede suceder porque su empresa puede bloquear conexiones a repositorios ampliamente utilizados en comunidades Python y JavaScript. Para obtener más información, consulte Installation Problems (Problemas de instalación) en la página web de JupyterLab.

- Si se encuentra ante un error de connection refused (conexión rechazada) al intentar conectar al punto de enlace de desarrollo, es posible que esté usando un punto de enlace de desarrollo que está desactualizado. Intente crear un nuevo punto de enlace de desarrollo y volver a conectarse.

Tutorial: uso de un bloc de notas de SageMaker con su punto de conexión de desarrollo

En AWS Glue, puede crear un punto de enlace de desarrollo y, a continuación, crear un bloc de notas de SageMaker para ayudar a desarrollar sus scripts de machine learning y ETL. Un bloc de notas de SageMaker es una instancia de computación de machine learning completamente administrado que ejecuta la aplicación de bloc de notas de Jupyter.

1. En la consola de AWS Glue, seleccione Puntos de enlace de desarrollo para ir a la lista de puntos de enlace de desarrollo.
2. Seleccione la casilla situada junto al nombre de un punto de enlace de desarrollo que desee utilizar y, en el menú Action (Acción), elija Create SageMaker notebook (Crear bloc de notas de SageMaker).
3. Rellene la página Create and configure a notebook (Crear y configurar un bloc de notas) como se indica a continuación:

   a. Escriba un nombre para el bloc de notas.

   b. En Attach to development endpoint (Asociar a un punto de enlace de desarrollo), verifique el punto de enlace de desarrollo.

   c. Elija crear un rol AWS Identity and Access Management (IAM).

      Se recomienda crear un rol. Si utiliza un rol existente, asegúrese de que tiene los permisos necesarios. Para obtener más información, consulte the section called “Paso 6: Crear una política de IAM para blocs de notas de SageMaker”.

   d. (Opcional) Elija una VPC, una subred y uno o varios grupos de seguridad.

e. (Opcional) Elija una clave de cifrado de AWS Key Management Service.

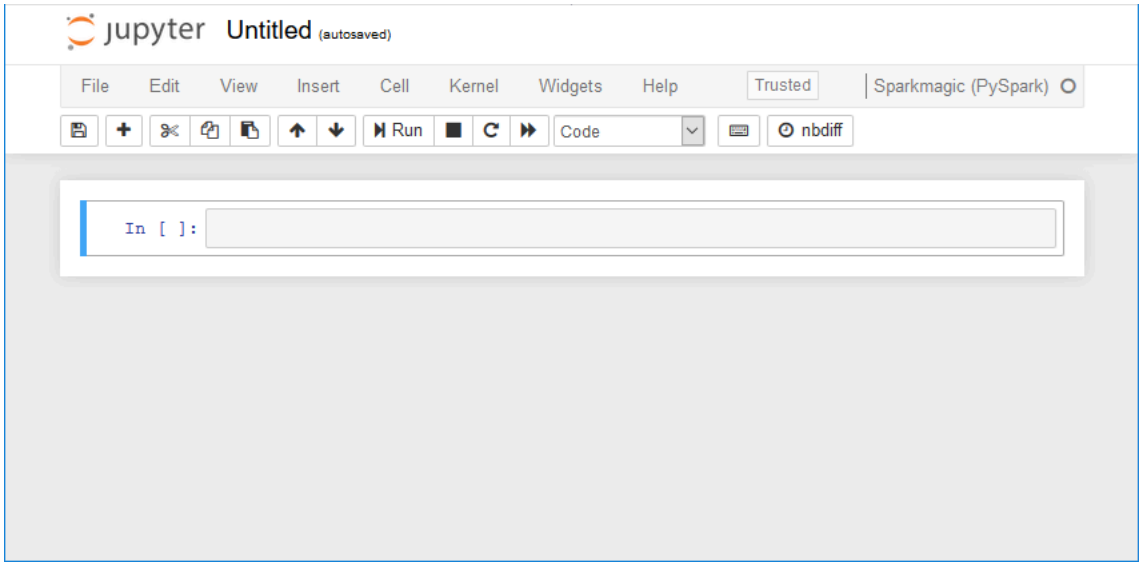
f. (Opcional) Añada etiquetas para la instancia de bloc de notas.

4. Elija Crear cuaderno. En la página Notebooks (Blocs de notas), elija el icono de actualización que aparece en la parte superior derecha y continúe hasta que el Status (Estado) aparezca como `Ready`.

5. Seleccione la casilla situada junto al nombre del bloc de notas nuevo y, a continuación, elija Open notebook (Abrir bloc de notas).

6. Cree un bloc de notas nuevo: en la página de jupyter, elija New (Nuevo) y, a continuación, seleccione Sparkmagic (PySpark).

   La pantalla que aparece debe ser similar a la siguiente:

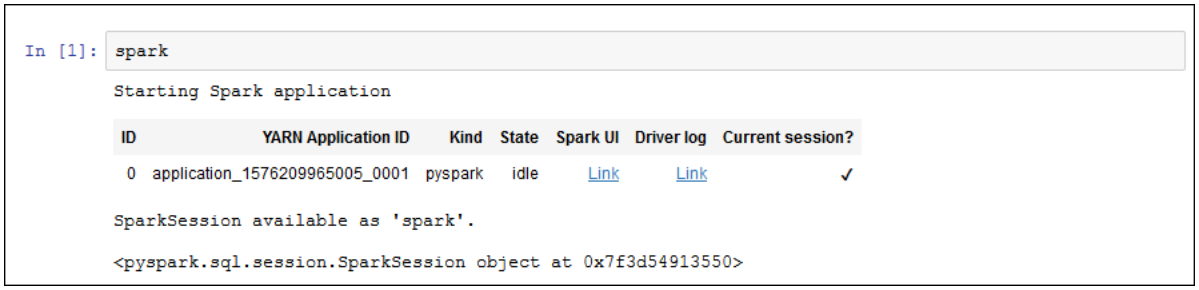


7. (Opcional) En la parte superior de la página, elija Untitled (Sin título) y asigne un nombre al bloc de notas.

8. Para iniciar una aplicación de Spark, escriba el comando siguiente en el bloc de notas y, a continuación, elija Run (Ejecutar) en la barra de herramientas.

```
spark
```

   Después de un breve intervalo, debería ver la respuesta siguiente:



9. Cree un marco dinámico y ejecute una consulta en él: copie, pegue y ejecute el código siguiente, que genera el recuento y el esquema de la tabla `persons_json`.

```
import sys
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.transforms import *
glueContext = GlueContext(SparkContext.getOrCreate())
persons_DyF = glueContext.create_dynamic_frame.from_catalog(database="legislators",
 table_name="persons_json")
print ("Count:  ", persons_DyF.count())
persons_DyF.printSchema()
```

Tutorial: Utilice un REPL Shell con su punto de conexión de desarrollo

En AWS Glue puede crear un punto de enlace de desarrollo y, a continuación, invocar un shell REPL (bucle de lectura, evaluación e impresión) para ejecutar código de PySpark gradualmente para depurar interactivamente sus scripts de ETL antes de implementarlos.

Para usar un REPL en un punto de conexión de desarrollo, necesita tener autorización para usar SSH en el punto de conexión.

1. En el equipo local, abra una ventana de terminal que pueda ejecutar comandos SSH y pegue en ella el comando SSH editado. Ejecute el comando .

   Suponiendo que aceptara la versión de AWS Glue 1.0 con Python 3 para el punto de conexión de desarrollo, el resultado será similar al siguiente:

```
Python 3.6.8 (default, Aug  2 2019, 17:42:44)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-28)] on linux
Type "help", "copyright", "credits" or "license" for more information.
SLF4J: Class path contains multiple SLF4J bindings.
```

```
SLF4J: Found binding in [jar:file:/usr/share/aws/glue/etl/jars/glue-assembly.jar!/
org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/lib/spark/jars/slf4j-log4j12-1.7.16.jar!/
org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use
 setLogLevel(newLevel).
2019-09-23 22:12:23,071 WARN  [Thread-5] yarn.Client (Logging.scala:logWarning(66))
 - Neither spark.yarn.jars nor spark.yarn.archive is set, falling back to uploading
 libraries under SPARK_HOME.
2019-09-23 22:12:26,562 WARN  [Thread-5] yarn.Client (Logging.scala:logWarning(66))
 - Same name resource file:/usr/lib/spark/python/lib/pyspark.zip added multiple
 times to distributed cache
2019-09-23 22:12:26,580 WARN  [Thread-5] yarn.Client (Logging.scala:logWarning(66))
 - Same path resource file:///usr/share/aws/glue/etl/python/PyGlue.zip added
 multiple times to distributed cache.
2019-09-23 22:12:26,581 WARN  [Thread-5] yarn.Client (Logging.scala:logWarning(66))
 - Same path resource file:///usr/lib/spark/python/lib/py4j-src.zip added multiple
 times to distributed cache.
2019-09-23 22:12:26,581 WARN  [Thread-5] yarn.Client (Logging.scala:logWarning(66))
 - Same path resource file:///usr/share/aws/glue/libs/pyspark.zip added multiple
 times to distributed cache.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 2.4.3
      /_/

Using Python version 3.6.8 (default, Aug  2 2019 17:42:44)
SparkSession available as 'spark'.
>>>
```

2. Compruebe que el shell REPL funcione correctamente escribiendo la instrucción `print(spark.version)`. Mientras esto muestre la versión de Spark, su REPL estará listo para su uso.
3. Ahora puede intentar ejecutar el siguiente script, que es sencillo, línea por línea, en el shell:

```
import sys
from pyspark.context import SparkContext
from awsglue.context import GlueContext
```

```
from awsglue.transforms import *
glueContext = GlueContext(SparkContext.getOrCreate())
persons_DyF = glueContext.create_dynamic_frame.from_catalog(database="legislators",
 table_name="persons_json")
print ("Count:  ", persons_DyF.count())
persons_DyF.printSchema()
```

Tutorial: Configuración de PyCharm Professional con un punto de conexión de desarrollo

En este tutorial, comprobará cómo conectar el IDE de Python PyCharm Professional que se ejecuta en su equipo local a un punto de enlace de desarrollo para que pueda ejecutar, depurar y probar de forma interactiva scripts de ETL (extraer, transferir y cargar) de AWS Glue antes de implementarlos. Las instrucciones y las capturas de pantalla del tutorial se basan en PyCharm Professional versión 2019.3.

Para conectarse a un punto de enlace de desarrollo de forma interactiva, debe haber instalado PyCharm Professional. Esto no podrá efectuarse con la edición gratuita.

> **Note**
>
> El tutorial utiliza Amazon S3 como origen de datos. Si desea utilizar un origen de datos JDBC en su lugar, debe ejecutar el punto de enlace de desarrollo en una nube virtual privada (VPC). Para conectarse con SSH a un punto de enlace de desarrollo en una instancia de VPC, debe crear un túnel SSH. Este tutorial no incluye instrucciones para crear túneles SSH. Para obtener información sobre el uso de SSH para conectarse a un punto de enlace de desarrollo en una VPC, consulte Securely Connect to Linux Instances Running in a Private Amazon VPC (Conexión segura a instancias de Linux que se ejecutan en una instancia de Amazon VPC privada) en el blog de seguridad de AWS.

Temas

- Conexión de PyCharm Professional a un punto de conexión de desarrollo
- Implementación del script en el punto de conexión de desarrollo
- Configuración de un intérprete remoto
- Ejecución del script en el punto de conexión de desarrollo

Conexión de PyCharm Professional a un punto de conexión de desarrollo

1. Cree un nuevo proyecto de Python puro en PyCharm y llámelo `legislators`.
2. Cree un archivo denominado `get_person_schema.py` en el proyecto con el siguiente contenido:

```
from pyspark.context import SparkContext
from awsglue.context import GlueContext


def main():
    # Create a Glue context
    glueContext = GlueContext(SparkContext.getOrCreate())

    # Create a DynamicFrame using the 'persons_json' table
    persons_DyF =
 glueContext.create_dynamic_frame.from_catalog(database="legislators",
 table_name="persons_json")

    # Print out information about this data
    print("Count:  ", persons_DyF.count())
    persons_DyF.printSchema()


if __name__ == "__main__":
    main()
```

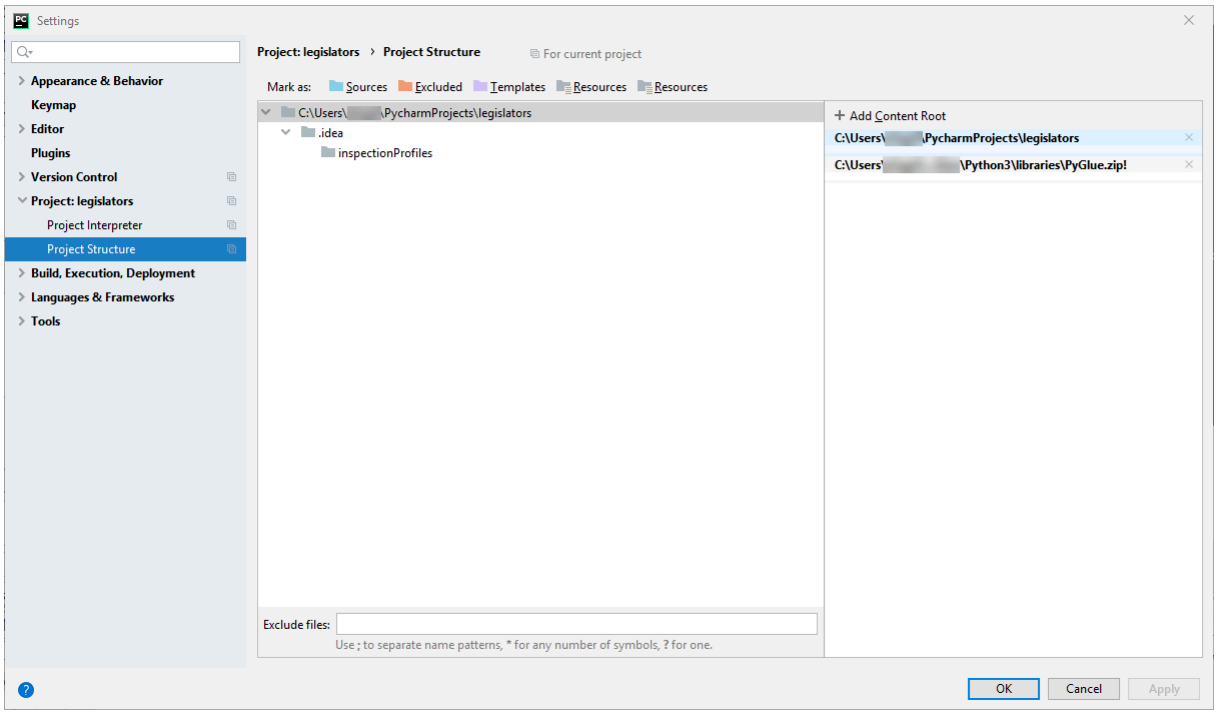
3. Realice una de las acciones siguientes:

   - Para la versión 0.9 de AWS Glue, descargue el archivo de la biblioteca de Python de `PyGlue.zip`, AWS Glue, desde `https://s3.amazonaws.com/aws-glue-jes-prod-us-east-1-assets/etl/python/PyGlue.zip` en la ubicación que prefiera de su equipo local.
   - Para la versión 1.0 y posteriores de AWS Glue, descargue el archivo de la biblioteca de Python de AWS Glue, `PyGlue.zip`, desde `https://s3.amazonaws.com/aws-glue-jes-prod-us-east-1-assets/etl-1.0/python/PyGlue.zip` en la ubicación que prefiera de su equipo local.

4. Añada `PyGlue.zip` como raíz de contenido para el proyecto en PyCharm:

- En PyCharm, seleccione File (Archivo) y Settings (Configuración) para abrir el cuadro de diálogo Settings (Configuración). (También puede pulsar `Ctrl+Alt+S`.)
- Amplíe el proyecto `legislators` y seleccione Project Structure (Estructura del proyecto). A continuación, en el panel derecho, seleccione + Add Content Root (+ Añadir raíz de contenido).
- Vaya a la ubicación donde guardó `PyGlue.zip`, selecciónelo y, a continuación, seleccione Apply (Aplicar).

La pantalla Settings (Configuración) debe asemejarse a la siguiente:



Deje abierto el cuadro de diálogo Settings (Configuración) después de seleccionar Apply (Aplicar).

5. Configure las opciones de implementación para cargar el script local en el punto de enlace de desarrollo con SFTP (esta función solo está disponible en PyCharm Professional):

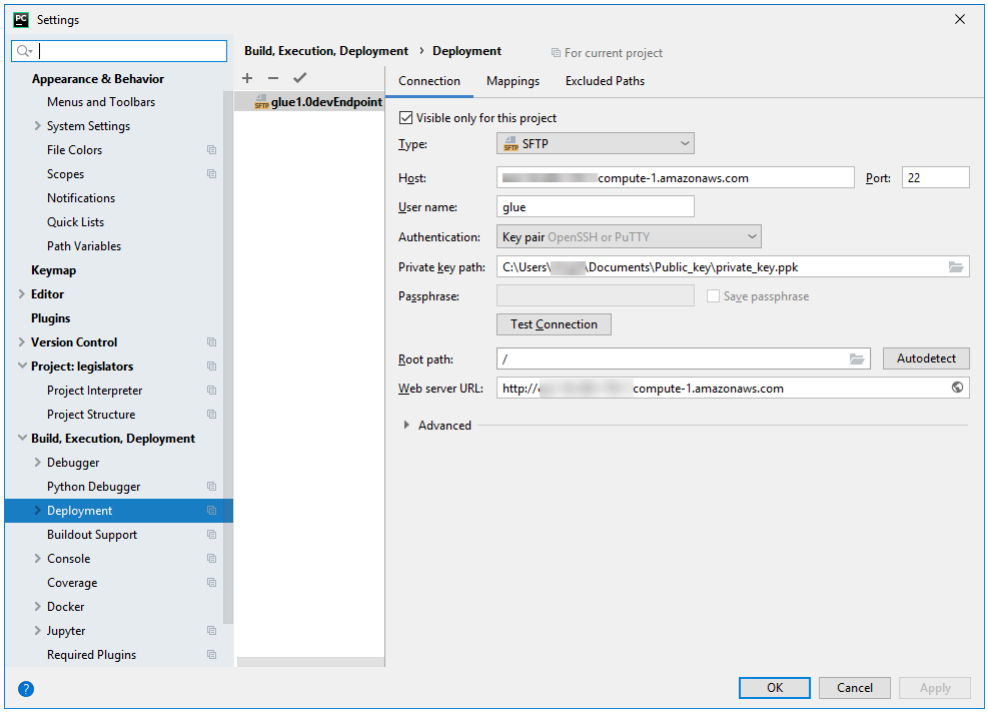
   - En el cuadro de diálogo Settings (Configuración), expanda la sección Build, Execution, Deployment (Generación, ejecución, implementación). Seleccione la subsección Deployment (Implementación).
   - Seleccione el icono + en la parte superior del panel central para añadir un nuevo servidor. Establezca Type (Tipo) en `SFTP` y asígnele un nombre.

- Establezca el SFTP host (Host SFTP) en la opción de Public address (Dirección pública) de su punto de enlace de desarrollo, tal y como aparece en la página de detalles. (Elija el nombre del punto de enlace de desarrollo en la consola de AWS Glue para mostrar la página de detalles). Para un punto de enlace de desarrollo que se ejecuta en una instancia de VPC, establezca el valor de SFTP host (Host SFTP) en la dirección del host y el puerto local del túnel SSH en el punto de enlace de desarrollo.
- Establezca User name (Nombre de usuario) en `glue`.
- Establezca Auth type (Tipo de autenticación) en Key pair (OpenSSH or Putty) (Par de claves (OpenSSH o Putty)). Defina Private key file (Archivo de clave privada); para ello, vaya a la ubicación donde se encuentra el archivo de clave privada del punto de enlace de desarrollo. Tenga en cuenta que PyCharm solo es compatible con los tipos de claves DSA, RSA y ECDSA OpenSSH y no acepta claves en formato privado de Putty. Puede utilizar una versión actualizada de `ssh-keygen` para generar un tipo de par de claves que acepte PyCharm, usando una sintaxis como la siguiente:

```
ssh-keygen -t rsa -f <key_file_name> -C "<your_email_address>"
```

- Seleccione Test connection (Probar conexión) y pruebe la conexión. Si la conexión se realiza correctamente, seleccione Apply (Aplicar).

La pantalla Configuración debe asemejarse ahora a la siguiente:

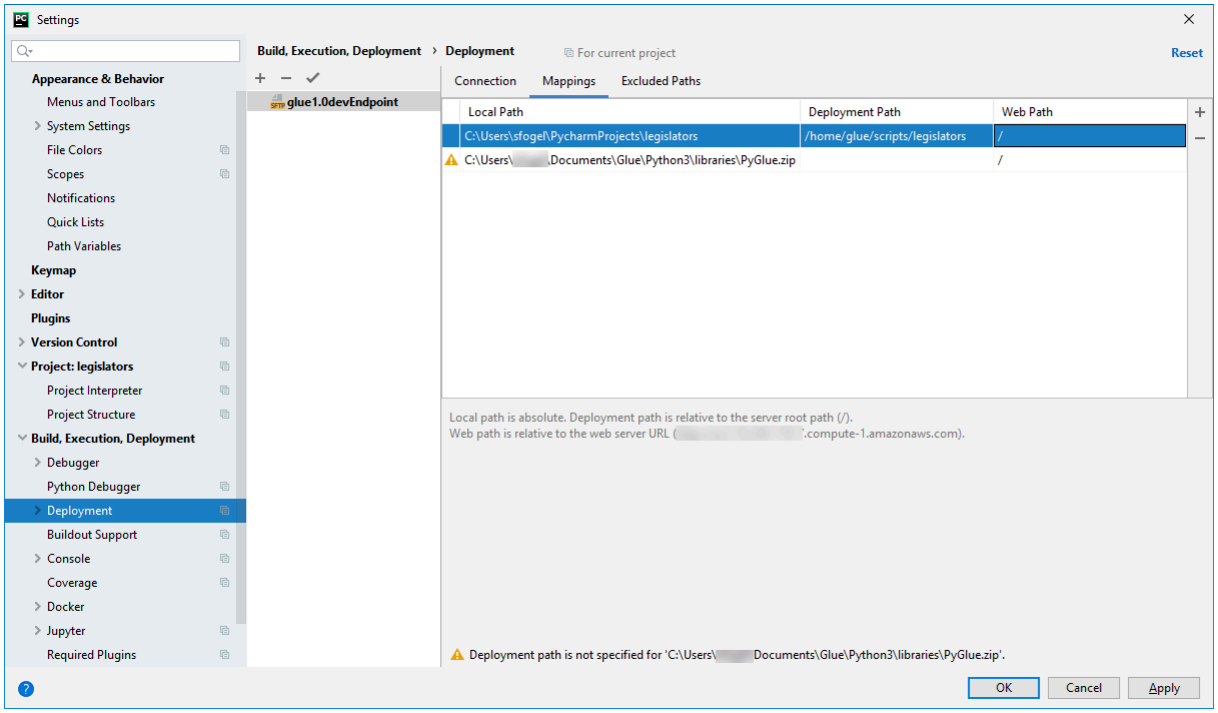


De nuevo, deje abierto el cuadro de diálogo Settings (Configuración) después de seleccionar Apply (Aplicar).

6. Asigne el directorio local a un directorio remoto para la implementación:

   - En el panel derecho de la página Deployment (Implementación), seleccione la pestaña central en la parte superior denominada Mappings (Mapeos).
   - En la columna Deployment Path (Ruta de la implementación), escriba una ruta en `/home/glue/scripts/` para la implementación de la ruta de su proyecto. Por ejemplo: `/home/glue/scripts/legislators`.
   - Seleccione Aplicar.

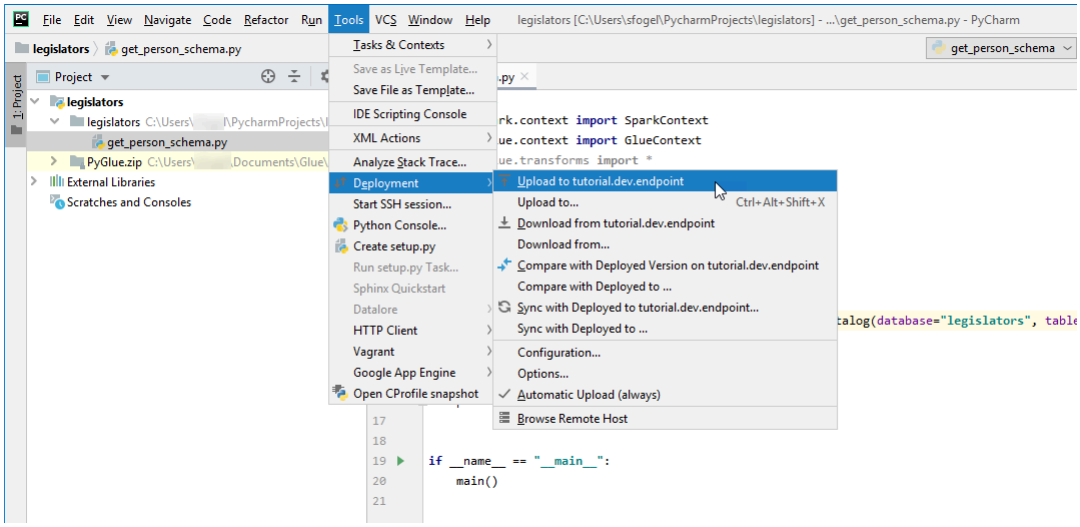
   La pantalla Configuración debe asemejarse ahora a la siguiente:



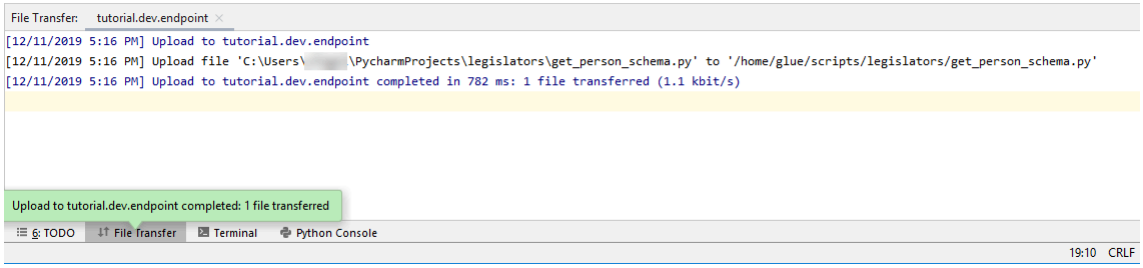
   Seleccione OK (Aceptar) para cerrar el cuadro de diálogo Settings (Configuración).

Implementación del script en el punto de conexión de desarrollo

1. Elija Tools (Herramientas), Deployment (Implementación) y, a continuación, seleccione el nombre con el que ha configurado el punto de enlace de desarrollo, tal como se muestra en la imagen siguiente:



Después de implementar el script, la parte inferior de la pantalla tendrá un aspecto similar al siguiente:



2. En la barra de menús, elija Tools (Herramientas), Deployment (Implementación), Automatic Upload (always) [Carga automática (siempre)]. Asegúrese de que aparezca una marca de verificación junto a Automatic Upload (always) [Carga automática (siempre)].

   Cuando esta opción está habilitada, PyCharm carga automáticamente los archivos modificados en el punto de enlace de desarrollo.

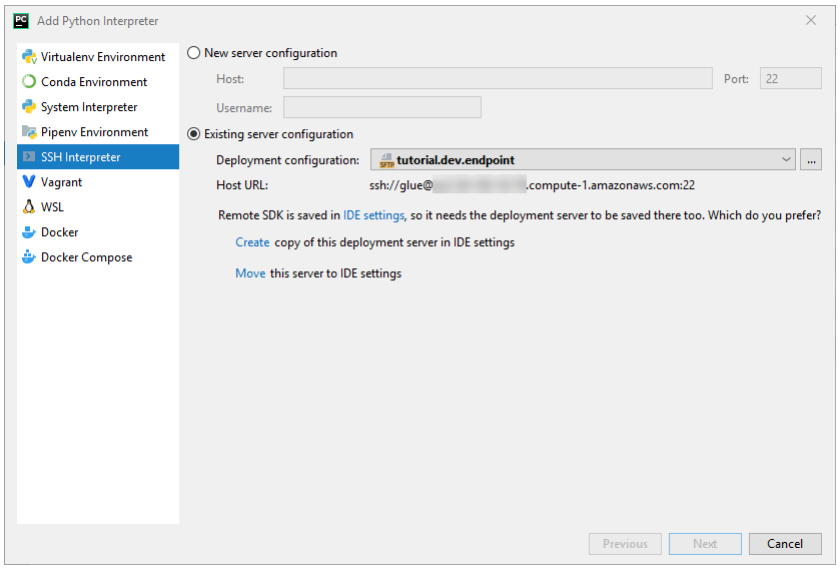
Configuración de un intérprete remoto

Configure PyCharm para utilizar el intérprete de Python en el punto de enlace de desarrollo.

1. En el menú File (Archivo), elija Settings (Configuración).
2. Expanda los legisladores del proyecto y elija Project Interpreter (Intérprete del proyecto).
3. Elija el icono de engranaje situado junto a la lista de Project Interpreter (Intérprete del proyecto) y, a continuación, elija Add (Agregar).
4. En el cuadro de diálogo Add Python Interpreter (Agregar intérprete de Python), en el panel izquierdo, elija SSH Interpreter (Intérprete de SSH).

5. Elija Existing server configuration (Configuración del servidor existente) y, en la lista Deployment configuration (Configuración de implementación), seleccione la configuración.

   La pantalla tendrá un aspecto similar a la imagen siguiente:



6. Elija Move this server to IDE settings (Mover este servidor a la configuración de IDE), y, a continuación, elija Next (Siguiente).
7. En el campo Interpreter (Intérprete), cambie la ruta a `/usr/bin/gluepython` si está utilizando Python 2 o a `/usr/bin/gluepython3` si está utilizando Python 3. A continuación, elija Finalizar.

Ejecución del script en el punto de conexión de desarrollo

Para ejecutar el script:

- En el panel izquierdo, haga clic con el botón derecho en el nombre del archivo y elija Run (Ejecutar) "***<nombreArchivo>***".

  Después de una serie de mensajes, la salida final debe mostrar el recuento y el esquema.

```
Count:  1961
root
|-- family_name: string
|-- name: string
|-- links: array
|    |-- element: struct
|    |    |-- note: string
```

```
|    |    |-- url: string
|-- gender: string
|-- image: string
|-- identifiers: array
|    |-- element: struct
|    |    |-- scheme: string
|    |    |-- identifier: string
|-- other_names: array
|    |-- element: struct
|    |    |-- lang: string
|    |    |-- note: string
|    |    |-- name: string
|-- sort_name: string
|-- images: array
|    |-- element: struct
|    |    |-- url: string
|-- given_name: string
|-- birth_date: string
|-- id: string
|-- contact_details: array
|    |-- element: struct
|    |    |-- type: string
|    |    |-- value: string
|-- death_date: string


Process finished with exit code 0
```

Ya ha terminado la configuración para depurar el script de forma remota en el punto de enlace de desarrollo.

Configuración avanzada: uso compartido de puntos de conexión de desarrollo entre múltiples usuarios

En esta sección se explica cómo puede aprovechar los puntos de desarrollo con los blocs de notas de SageMaker en casos de uso habituales para compartir puntos de enlace de desarrollo entre múltiples usuarios.

Configuración de tenencia única

En casos de uso de tenencia única, a fin de simplificar la experiencia del desarrollador y evitar la contención de recursos, se recomienda que cada desarrollador use su propio punto de enlace

de desarrollo dimensionado para el proyecto en el que está trabajando. Esto también simplifica las decisiones relacionadas con el tipo de empleado y el recuento de DPU y las deja a discreción del desarrollador y según el proyecto en el que están trabajando.

No tendrá que ocuparse de la asignación de recursos a menos que ejecute múltiples archivos de bloc de notas en forma concurrente. Si ejecuta código en varios archivos de bloc de notas al mismo tiempo, se iniciarán múltiples sesiones de Livy en forma concurrente. Para segregar las configuraciones del clúster de Spark con el fin de ejecutar múltiples sesiones de Livy al mismo tiempo, puede seguir los pasos que se presentan en los casos de uso de multiinquilinos.

Por ejemplo, si el punto de enlace de desarrollo tiene 10 empleados y el tipo de empleado es `G.1X`, entonces tendrá 9 ejecutores de Spark y todo el clúster tendrá 90 G de memoria del ejecutor ya que cada ejecutor tendrá 10 G de memoria.

Independientemente del tipo de empleado especificado, se activará la asignación dinámica de recursos de Spark. Si un conjunto de datos es lo suficientemente grande, Spark puede asignar todos los ejecutores a una única sesión de Livy ya que `spark.dynamicAllocation.maxExecutors` no está configurado en forma predeterminada. Esto significa que otras sesiones de Livy en el mismo punto de enlace de desarrollo esperarán para lanzar nuevos ejecutores. Si el conjunto de datos es pequeño, Spark podrá asignar ejecutores a múltiples sesiones de Livy al mismo tiempo.

> **Note**
>
> Para obtener más información acerca de cómo se asignan los recursos en diferentes casos de uso y cómo se establece una configuración para modificar el comportamiento, consulte Configuración avanzada: uso compartido de puntos de conexión de desarrollo entre múltiples usuarios.

Configuración de multiinquilinos

> **Note**
>
> Tenga en cuenta que los puntos de enlace de desarrollo están destinados a emular el entorno ETL de AWS Glue como un entorno de tenencia única. Si bien el uso de multiinquilinos es posible, es un caso de uso avanzado y se recomienda que la mayoría de los usuarios mantengan un patrón de tenencia única para cada punto de enlace de desarrollo.

En casos de uso de multiinquilinos, es posible que tenga que ocuparse de la asignación de recursos. El factor clave es el número de usuarios concurrentes que usan un bloc de notas Jupyter al mismo tiempo. Si su equipo trabaja en un flujo de trabajo de “seguir el sol” y solo hay un usuario de Jupyter en cada zona horaria, entonces el número de usuarios concurrentes es solo uno, por lo que no tendrá que preocuparse por la asignación de recursos. Sin embargo, si su bloc de notas se comparte entre varios usuarios y cada usuario envía código de forma ad-hoc, tendrá que considerar los siguientes puntos.

Para particionar los recursos del clúster de Spark entre varios usuarios, puede usar configuraciones de SparkMagic. Existen dos formas diferentes de configurar SparkMagic.

(A) Use la directiva %%configure -f

Si desea modificar la configuración por cada sesión de Livy desde el bloc de notas, puede ejecutar la directiva `%%configure -f` en el párrafo del bloc de notas.

Por ejemplo, si desea ejecutar la aplicación Spark en 5 ejecutores, puede ejecutar el siguiente comando en el párrafo del bloc de notas.

```
%%configure -f
{"numExecutors":5}
```

Luego verá solo 5 ejecutores ejecutándose para el trabajo en la interfaz de usuario de Spark.

Recomendamos limitar el número máximo de ejecutores para la asignación dinámica de recursos.

```
%%configure -f
{"conf":{"spark.dynamicAllocation.maxExecutors":"5"}}
```

(B) Modificar el archivo de configuración de SparkMagic

SparkMagic funciona con base en la API de Livy. SparkMagic crea sesiones de Livy con configuraciones como `driverMemory`, `driverCores`, `executorMemory`, `executorCores`, `numExecutors`, `conf`, etc. Estos son los factores clave que determinan la cantidad de recursos que se consumen de todo el clúster de Spark. SparkMagic le permite proporcionar un archivo de configuración para especificar los parámetros que se envían a Livy. Puede ver un archivo de configuración de ejemplo en este Repositorio GitHub.

Si desea modificar la configuración en todas las sesiones de Livy desde un bloc de notas, puede modificar `/home/ec2-user/.sparkmagic/config.json` para agregar `session_config`.

Para modificar el archivo de configuración en una instancia de bloc de notas de SageMaker, puede seguir estos pasos.

1. Abra un bloc de notas de SageMaker.
2. Abra el kernel del terminal.
3. Ejecute los comandos siguientes:

```
sh-4.2$ cd .sparkmagic
sh-4.2$ ls
config.json logs
sh-4.2$ sudo vim config.json
```

Por ejemplo, puede agregar estas líneas a `/home/ec2-user/.sparkmagic/config.json` y reiniciar el kernel de Jupyter desde el bloc de notas.

```
  "session_configs": {
    "conf": {
      "spark.dynamicAllocation.maxExecutors":"5"
    }
  },
```

Directrices y prácticas recomendadas

Para evitar este tipo de conflicto de recursos, puede usar algunos enfoques básicos como por ejemplo:

- Tener un clúster de Spark más grande al aumentar el `NumberOfWorkers` (escalar horizontalmente) y actualizar el `workerType` (escalar verticalmente)
- Asignar menos recursos por usuario (menos recursos por sesión de Livy)

Su enfoque dependerá del caso de uso. Si tiene un punto de enlace de desarrollo más grande y no hay una gran cantidad de datos, la posibilidad de un conflicto de recursos disminuirá significativamente porque Spark puede asignar recursos en función de una estrategia de asignación dinámica.

Como se describió con anterioridad, el número de ejecutores de Spark puede calcularse automáticamente con base en una combinación de DPU (o `NumberOfWorkers`) y tipo de empleado. Cada aplicación Spark lanza un controlador y varios ejecutores. Para calcular necesitará

`NumberOfWorkers` = `NumberOfExecutors + 1`. La siguiente matriz explica cuánta capacidad necesita en su punto de enlace de desarrollo en función del número de usuarios concurrentes.

Número de usuarios concurrentes del bloc de notas	Número de ejecutores de Spark que desea asignar por usuario	Número total de empleados para su punto de enlace de desarrollo
3	5	18
10	5	60
50	5	300

Si desea asignar menos recursos por usuario, `spark.dynamicAllocation.maxExecutors` (o `numExecutors`) sería el parámetro más fácil de configurar como parámetro de sesión de Livy. Si establece la siguiente configuración en `/home/ec2-user/.sparkmagic/config.json`, entonces SparkMagic asignará un máximo de 5 ejecutores por sesión de Livy. Esto ayudará a segregar recursos por sesión de Livy.

```
"session_configs": {
    "conf": {
      "spark.dynamicAllocation.maxExecutors":"5"
    }
  },
```

Supongamos que hay un punto de enlace de desarrollo con 18 empleados (G-1 X) y que hay tres usuarios de bloc de notas concurrentes. Si la configuración de su sesión tiene `spark.dynamicAllocation.maxExecutors=5` entonces cada usuario puede hacer uso de un controlador y cinco ejecutores. No habrá ningún conflicto de recursos incluso cuando ejecute varios párrafos de bloc de notas al mismo tiempo.

Compensación

Con esta configuración de sesión `"spark.dynamicAllocation.maxExecutors":"5"`, podrá evitar errores de conflictos de recursos y no necesitará esperar a la asignación de recursos cuando haya accesos de usuario concurrentes. Sin embargo, incluso cuando haya muchos recursos libres

(por ejemplo, no hay otros usuarios concurrentes), Spark no puede asignar más de 5 ejecutores para su sesión de Livy.

Otras notas

Es una buena práctica detener el kernel de Jupyter cuando deje de usar un bloc de notas. Esto liberará recursos y otros usuarios de bloc de notas podrán usar esos recursos inmediatamente sin esperar al vencimiento del kernel (apagado automático).

Problemas comunes

Es posible que experimentes ciertos problemas, aún cuando siga las pautas.

No se encuentra la sesión

Cuando intenta ejecutar un párrafo de bloc de notas aunque su sesión de Livy ya ha sido finalizada, verá el siguiente mensaje. Para habilitar la sesión de Livy, es necesario reiniciar el kernel de Jupyter al seleccionar Kernel > Restart (Reiniciar) en el menú Jupyter y, a continuación, vuelva a ejecutar el párrafo del bloc de notas.

```
An error was encountered:
Invalid status code '404' from http://localhost:8998/sessions/13 with error payload:
 "Session '13' not found."
```

No hay recursos suficientes de YARN

Cuando intenta ejecutar un párrafo de bloc de notas aunque su clúster de Spark no tenga suficientes recursos para iniciar una nueva sesión de Livy, verá el siguiente mensaje. En general, este problema se puede evitar si sigue las pautas; sin embargo, existe la posibilidad de que el problema surja. Para solucionar el problema, puede comprobar si hay alguna sesión de Livy activa que no sea necesaria. Si hay sesiones de Livy innecesarias, tendrá que finalizarlas para liberar los recursos del clúster. Para obtener más información, consulte la siguiente sección.

```
Warning: The Spark session does not have enough YARN resources to start.
The code failed because of a fatal error:
    Session 16 did not start up in 60 seconds..

Some things to try:
a) Make sure Spark has enough available resources for Jupyter to create a Spark
 context.
```

```
b) Contact your Jupyter administrator to make sure the Spark magics library is
 configured correctly.
c) Restart the kernel.
```

Monitoreo y depuración

En esta sección se describen las técnicas para monitorear los recursos y las sesiones.

Monitoreo y depuración de la asignación de recursos de clúster

Puede ver la interfaz de usuario de Spark para monitorear cuántos recursos se asignan por sesión de Livy y cuáles son las configuraciones efectivas de Spark mientras trabaja. Para habilitar la interfaz de usuario de Spark, consulte Habilitación de la interfaz de usuario web de Apache Spark para puntos de enlace de desarrollo.

(Opcional) si necesita una vista en tiempo real de la interfaz de usuario de Spark, puede configurar un túnel SSH en relación con el servidor de historial de Spark que se ejecuta en el clúster de Spark.

```
ssh -i <private-key.pem> -N -L 8157:<development endpoint public address>:18080
 glue@<development endpoint public address>
```

Luego, puede abrir http://localhost:8157 en su navegador para ver la interfaz de usuario de Spark.

Sesiones de Livy libres innecesarias

Revise estos procedimientos para cerrar las sesiones de Livy innecesarias desde un bloc de notas o un clúster de Spark.

(a). Finalizar sesiones de Livy desde un bloc de notas

Puede apagar el kernel en un bloc de notas de Jupyter para finalizar las sesiones de Livy innecesarias.

(b). Finalizar sesiones de Livy desde un clúster de Spark

Si hay sesiones de Livy innecesarias que todavía se están ejecutando, puede cerrar las sesiones de Livy en el clúster de Spark.

Como requisito previo para realizar este procedimiento, debe configurar la clave pública SSH para el punto de enlace de desarrollo.

Para iniciar sesión en el clúster de Spark, puede ejecutar el siguiente comando:

```
$ ssh -i <private-key.pem> glue@<development endpoint public address>
```

Puede ejecutar el siguiente comando para ver las sesiones activas de Livy:

```
$ yarn application -list
20/09/25 06:22:21 INFO client.RMProxy: Connecting to ResourceManager at
 ip-255-1-106-206.ec2.internal/172.38.106.206:8032
Total number of applications (application-types: [] and states: [SUBMITTED, ACCEPTED,
 RUNNING]):2
Application-Id Application-Name Application-Type User Queue State Final-State Progress
 Tracking-URL
application_1601003432160_0005 livy-session-4 SPARK livy default RUNNING UNDEFINED 10%
 http://ip-255-1-4-130.ec2.internal:41867
application_1601003432160_0004 livy-session-3 SPARK livy default RUNNING UNDEFINED 10%
 http://ip-255-1-179-185.ec2.internal:33727
```

Puede cerrar la sesión de Livy con el siguiente comando:

```
$ yarn application -kill application_1601003432160_0005
20/09/25 06:23:38 INFO client.RMProxy: Connecting to ResourceManager at
 ip-255-1-106-206.ec2.internal/255.1.106.206:8032
Killing application application_1601003432160_0005
20/09/25 06:23:39 INFO impl.YarnClientImpl: Killed application
 application_1601003432160_0005
```

Administración de blocs de notas

> **Note**
>
> Los puntos de conexión de desarrollo solo son compatibles con versiones de AWS Glue anteriores a la 2.0. Si busca un entorno interactivo en el que pueda crear y probar scripts de ETL, utilice los Blocs de notas en AWS Glue Studio.

Un cuaderno permite tareas de desarrollo y pruebas interactivas de sus scripts de ETL (extracción, transformación y carga) en un punto de conexión de desarrollo. AWS Glue proporciona una interfaz para cuadernos de Jupyter SageMaker. Con AWS Glue, el usuario crea y administra blocs de notas de SageMaker. También puede abrir blocs de notas de SageMaker desde la consola de AWS Glue.

Además, puede utilizar Apache Spark con SageMaker en puntos de enlace de desarrollo de AWS Glue que soportan SageMaker (pero no trabajos de ETL de AWS Glue). SageMaker Spark es una biblioteca Apache Spark de código abierto para SageMaker. Para obtener más información, consulte Uso de Apache Spark con Amazon SageMaker.

Important

La administración de blocs de notas de SageMaker con los puntos de enlace de desarrollo de AWS Glue está disponible en las siguientes regiones de AWS:

Región	Código
US East (Ohio)	`us-east-2`
Este de EE. UU. (Norte de Virginia)	`us-east-1`
Oeste de EE. UU. (Norte de California)	`us-west-1`
Oeste de EE. UU. (Oregón)	`us-west-2`
Asia-Pacífico (Tokio)	`ap-northeast-1`
Asia-Pacífico (Seúl)	`ap-northeast-2`
Asia-Pacífico (Mumbai)	`ap-south-1`
Asia-Pacífico (Singapur)	`ap-southeast-1`
Asia-Pacífico (Sídney)	`ap-southeast-2`
Canadá (centro)	`ca-central-1`
Europa (Fráncfort)	`eu-central-1`
Europa (Irlanda)	`eu-west-1`
Europa (Londres)	`eu-west-2`

Creación de trabajos de ETL visuales con AWS Glue Studio

Un trabajo de AWS Glue encapsula un script que se conecta a los datos de origen, los procesa y, a continuación, los escribe en el destino de datos. Normalmente, un trabajo ejecuta scripts de extracción, transformación y carga (ETL). Los trabajos pueden ejecutar scripts diseñados para los entornos de tiempo de ejecución de Apache Spark y Ray. Los trabajos también pueden ejecutar scripts de Python de uso general (trabajos de shell de Python). Los desencadenadores de AWS Glue pueden iniciar trabajos en función de un programa o evento, o bajo demanda. Puede monitorear las ejecuciones de trabajos para comprender las métricas de tiempo de ejecución como el estado de realización, la duración y la hora de inicio.

Puede utilizar los scripts que genera AWS Glue o puede utilizar sus propios scripts. Si se cuenta con un esquema de origen y una ubicación o esquema de destino determinados, el generador de código de AWS Glue Studio puede crear automáticamente un script de la API de Apache Spark (PySpark). Puede utilizar este script como base y editarlo para satisfacer sus objetivos.

AWS Glue puede escribir archivos de salida en varios formatos de datos. Cada tipo de trabajo puede admitir diferentes formatos de salida. En algunos formatos de datos, se pueden escribir formatos de compresión comunes.

Inicie sesión en la consola de AWS Glue

Un flujo de trabajo en AWS Glue es la lógica empresarial que lleva a cabo el flujo de trabajo de extracción, transformación y carga (ETL). Puede crear trabajos en la sección ETL de la consola de AWS Glue.

Para ver los trabajos existentes, inicie sesión en la AWS Management Console y abra la consola de AWS Glue en https://console.aws.amazon.com/glue/. Después, seleccione pestaña Jobs (Trabajos) en AWS Glue. En la lista Jobs (Trabajos) se muestra la ubicación del script que se asocia a cada flujo de trabajo, cuando el flujo de trabajo se modificó por última vez, y la opción de marcador de flujo de trabajo actual.

Al crear un nuevo trabajo o después de haberlo guardado, puede utilizar AWS Glue Studio para modificar los trabajos de ETL. Puede hacerlo mediante la edición de los nodos en el editor visual o al editar el script de trabajo en modo desarrollador. También puede agregar y eliminar nodos en el editor visual para crear trabajos de ETL más complejos.

Siguientes pasos para crear un trabajo en AWS Glue Studio

Utilice el editor visual de trabajos para configurar nodos para su trabajo. Cada nodo representa una acción, como leer datos de la ubicación de origen o aplicar una transformación a los datos. Cada nodo que agregue al trabajo tiene propiedades que proporcionan información sobre la ubicación o la transformación de los datos.

Los próximos pasos para crear y administrar sus trabajos son los siguientes:

- Operaciones de ETL visuales con AWS Glue Studio
- Ver el script de trabajo
- Modificar las propiedades del trabajo
- Guardar el trabajo
- Iniciar una ejecución de trabajo
- Ver información sobre las ejecuciones de trabajos recientes
- Acceso al panel de monitoreo de trabajos

Operaciones de ETL visuales con AWS Glue Studio

Puede utilizar la interfaz visual simple en AWS Glue Studio para crear los trabajos de ETL. Se utiliza la página Jobs (Trabajos) para crear nuevos trabajos. También puede utilizar un editor de script o bloc de notas para trabajar en forma directa con el código en el script del trabajo de ETL de AWS Glue Studio.

En la página Jobs (Trabajos), puede ver todos los trabajos que ha creado con AWS Glue Studio o AWS Glue. Puede ver, administrar y ejecutar sus trabajos en esta página.

Consulte también el tutorial del blog para ver otro ejemplo de cómo crear trabajos de ETL con AWS Glue Studio.

Empezar trabajos en AWS Glue Studio

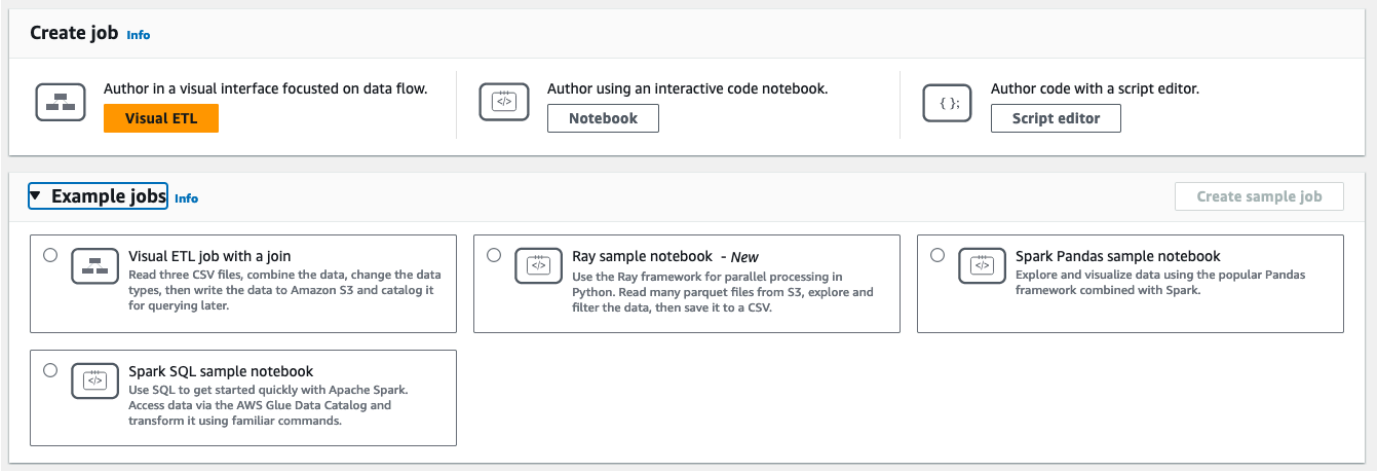
AWS Glue permite crear un trabajo a través de una interfaz visual, un cuaderno de códigos interactivo o con un editor de guiones. Puede iniciar un trabajo al hacer clic en cualquiera de las opciones o crear uno nuevo a partir de un trabajo de muestra.

Los trabajos de muestra crean un trabajo con la herramienta que usted elija. Por ejemplo, los trabajos de muestra permiten crear un trabajo ETL visual que combine archivos CSV en una tabla de

catálogos, crear un trabajo en un cuaderno de códigos interactivo con AWS Glue para Ray o AWS Glue para Spark cuando trabaje con pandas o crear un trabajo en un cuaderno de códigos interactivo con SparkSQL.

Creación de un trabajo en AWS Glue Studio desde cero

1. Inicie sesión en la AWS Glue Studio consola AWS Management Console y ábrala en https://console.aws.amazon.com/gluestudio/.
2. En el panel de navegación, seleccione Trabajos ETL.
3. En la sección Crear trabajo, seleccione una opción de configuración para el trabajo.



Opciones para crear un trabajo desde cero:

- Visual ETL: autor en una interfaz visual centrada en el flujo de datos
- Autor mediante cuaderno de códigos interactivo: cree trabajos de forma interactiva en una interfaz de cuaderno basada en cuadernos de Jupyter

  Cuando selecciona esta opción, debe proporcionar información adicional antes de crear una sesión de creación de un cuaderno. Para conocer más acerca de cómo especificar esta información, consulte Introducción a los cuaderno en AWS Glue Studio.
- Autor de código con un editor de guiones: para aquellos que estén familiarizados con la programación y la escritura de guiones de ETL, elija esta opción a fin de crear un nuevo trabajo ETL de Spark. Elige el motor (Python shell, Ray, Spark (Python) o Spark (Scala). A continuación, seleccione Comenzar de cero o Cargar guion para cargar un guion existente desde un archivo local. Si elige utilizar el editor de script, no puede usar el editor visual para diseñar o editar el trabajo.

Un trabajo de Spark se ejecuta en un entorno Apache Spark administrado por AWS Glue. De forma predeterminada, los nuevos scripts están codificados en Python. Para escribir un nuevo script de Scala, consulte Creación y edición de scripts de Scala en AWS Glue Studio.

Creación de un trabajo en AWS Glue Studio a partir de un ejemplo de trabajo

Puede optar por crear un trabajo a partir de un trabajo de muestra. En Trabajos de ejemplo, seleccione un trabajo de ejemplo y, a continuación, seleccione Crear un trabajo de muestra para iniciar un trabajo de muestra. Al crear un trabajo de muestra a partir de una de las opciones, se proporciona una plantilla rápida con la que puede trabajar.

1. Inicie sesión en la AWS Glue Studio consola AWS Management Console y ábrala en https://console.aws.amazon.com/gluestudio/.
2. En el panel de navegación, seleccione Trabajos ETL.
3. Seleccione una opción para crear un trabajo a partir de un trabajo de muestra:
   - Trabajo de ETL visual para unir varios orígenes: lea tres archivos CSV, combine los datos, cambie los tipos de datos y, a continuación, escriba los datos en Amazon S3 y catalóguelos para consultarlos más adelante.
   - Cuaderno Spark con Pandas: explore y visualice datos con el marco popular de Pandas combinado con Spark.
   - Cuaderno Spark con SQL: use SQL para empezar rápidamente con Apache Spark. Acceda a los datos a través del catálogo de datos de AWS Glue y transfórmelos con comandos conocidos.
4. Seleccione Crear un trabajo de muestra.

Características del editor de trabajo

El editor de trabajos proporciona las siguientes características para crear y editar trabajos.

- Diagrama visual de su trabajo, con un nodo para cada tarea de trabajo: nodos de origen de datos para leer los datos; nodos de transformación para modificar los datos; nodos de destino de datos para escribir los datos.

  Puede ver y configurar las propiedades de cada nodo en el diagrama de trabajo. También puede ver el esquema y los datos de ejemplo de cada nodo en el diagrama de trabajo. Estas

características ayudan a comprobar que su trabajo está modificando y transformando los datos de la manera correcta, sin tener que ejecutar el trabajo.

- Una pestaña Visualización y edición de scripts, donde puede modificar el código generado para su trabajo.
- Una pestaña Detalles del trabajo, en la que puede configurar una variedad de opciones para personalizar el entorno en el que se ejecuta el trabajo de ETL de AWS Glue.
- Una pestaña Ejecuciones, donde puede ver las ejecuciones actuales y anteriores del trabajo, ver el estado de la ejecución del trabajo y acceder a los registros de la ejecución del trabajo.
- Una pestaña de calidad de datos, donde puede aplicar reglas de calidad de datos a su trabajo.
- Una pestaña Programaciones, en la que puede configurar la hora de inicio del trabajo o configurar ejecuciones de trabajos recurrentes.
- Una pestaña de Control de versiones, donde puede configurar un servicio de Git para usarlo en el trabajo.

Uso de previsualizaciones de esquema en el editor visual de trabajos

Mientras crea o edita su trabajo, puede usar la pestaña Esquema de salida para ver el esquema de sus datos.

Antes de ver el esquema, el editor de trabajos necesita permisos para acceder al origen de datos. Puede especificar un rol de IAM en la pestaña Detalles del trabajo del editor o en la pestaña Esquema de salida para un nodo. Si el rol de IAM tiene todos los permisos necesarios para acceder al origen de datos, puede ver el esquema en la pestaña Esquema de salida para un nodo.

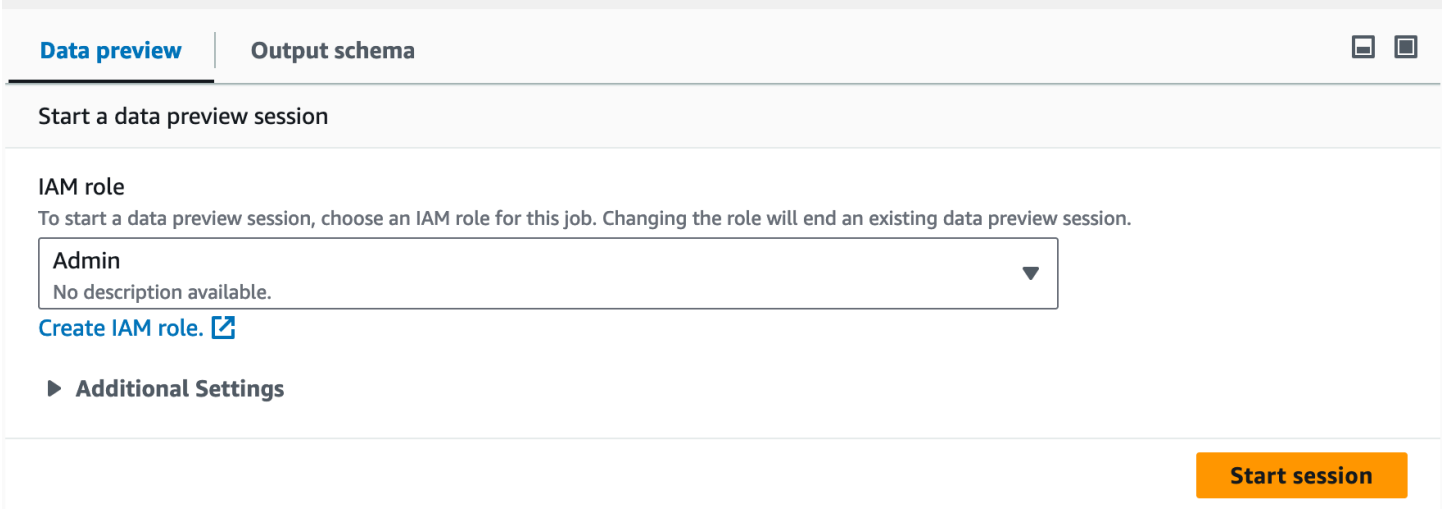
Uso de previsualizaciones de datos en el editor visual de trabajos

Las previsualizaciones de datos ayudan a crear y probar su trabajo con una muestra de sus datos, sin tener que ejecutarlo varias veces. Al utilizar la vista previa de datos, puede:

- Probar un rol de IAM para asegurarse de que tiene acceso a sus orígenes de datos o destinos de datos.
- Comprobar que la transformación está modificando los datos de la forma deseada. Por ejemplo, si utiliza una transformación de filtro, puede asegurarse de que el filtro está seleccionando el subconjunto correcto de datos.

- Comprobar sus datos. Si el conjunto de datos contiene columnas con valores de varios tipos, la previsualización de datos muestra una lista de tuplas para estas columnas. Cada tupla contiene el tipo de datos y su valor.

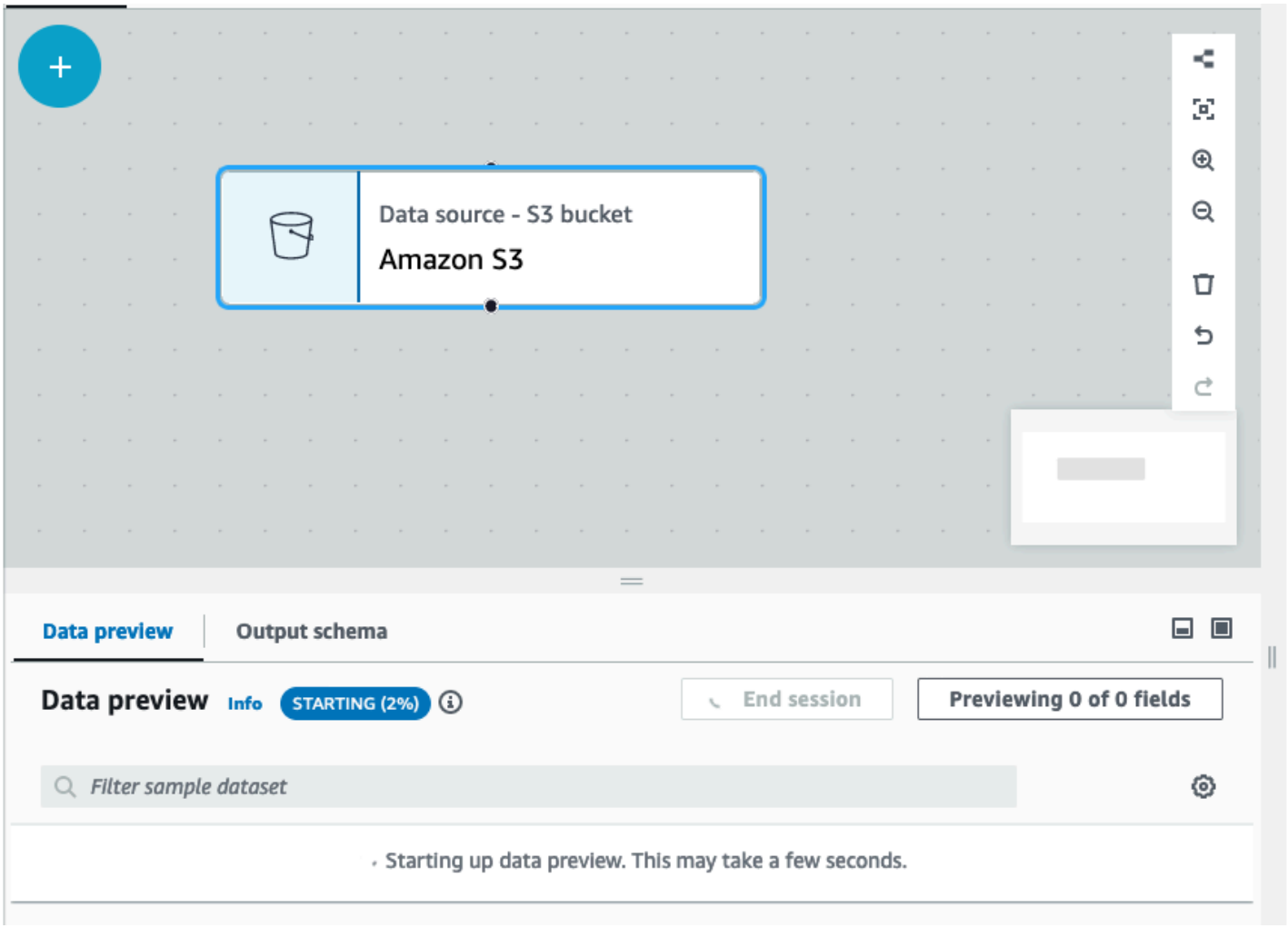
Mientras crea o edita el trabajo, puede usar la pestaña Previsualización de datos debajo del lienzo del trabajo para ver una muestra de sus datos. Se iniciará automáticamente una nueva sesión de vista previa de datos cuando el rol ya esté configurado en el trabajo o cuando se haya configurado un rol de IAM predeterminado en la cuenta. Si un rol no se ha configurado previamente, puede iniciar una sesión seleccionándolo.



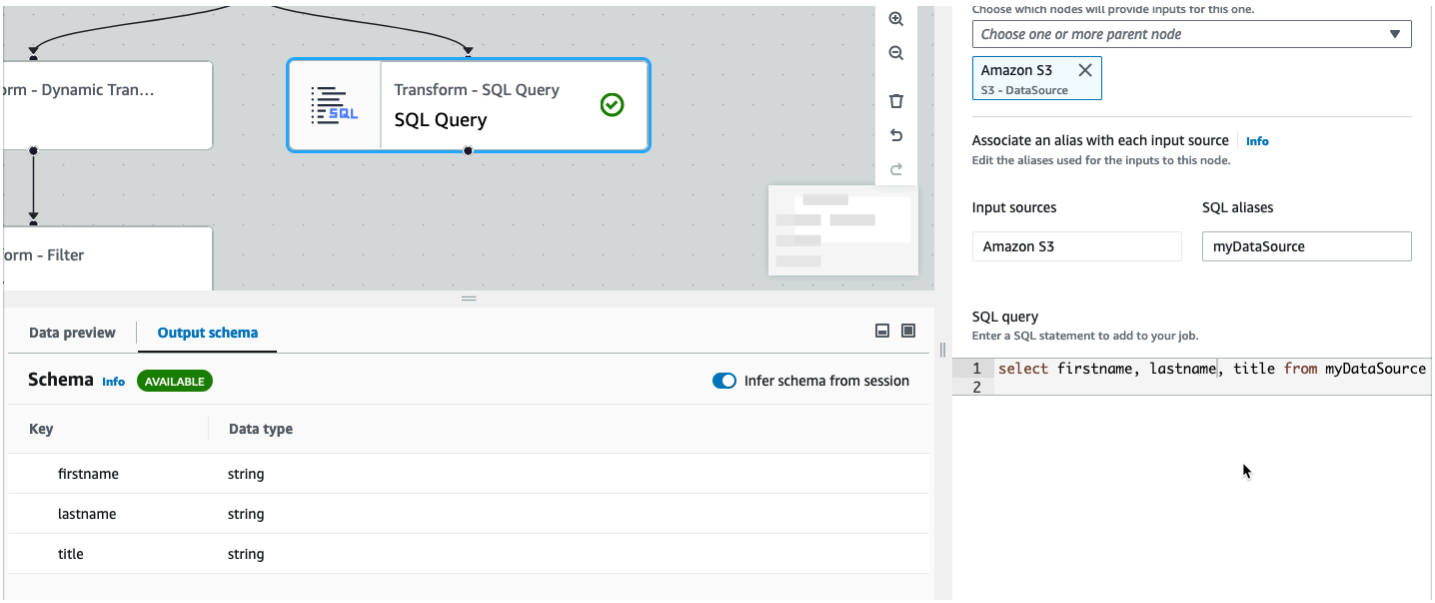
> **Note**
>
> El rol que elija para la sesión de vista previa de datos también se usará para el trabajo.

Para ver el estado y el progreso de la sesión, así como los detalles de la sesión, haga clic en el icono de información.

Cuando la sesión esté lista, AWS Glue Studio cargará los datos del nodo que seleccionó. Puede ver el porcentaje completado a medida que avanza.



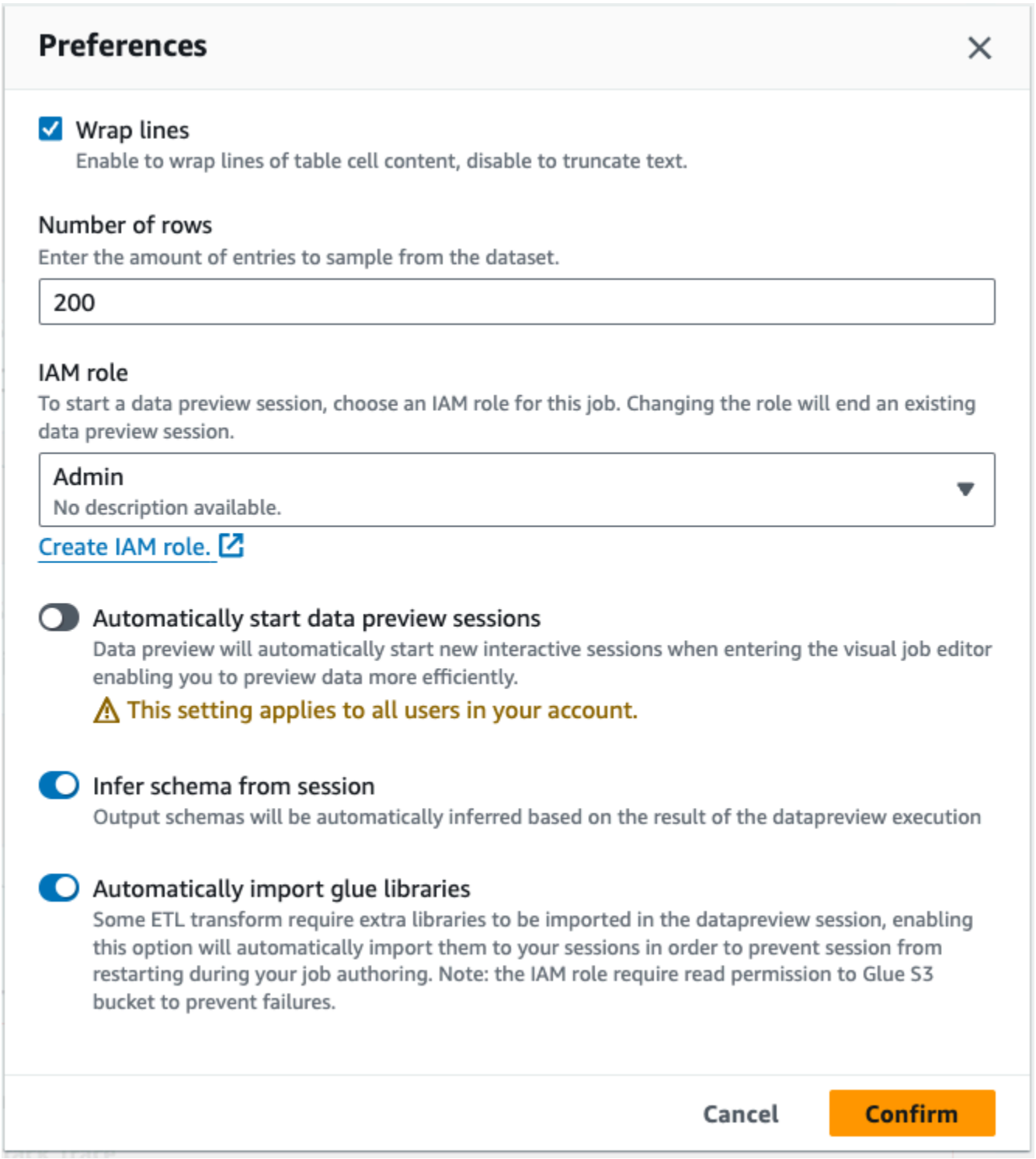
A medida que vaya creando su trabajo visual, AWS Glue Studio actualizará automáticamente el esquema del nodo seleccionado cuando active la opción Deducir el esquema de la sesión en la pestaña Esquema de salida.



Para configurar sus preferencias de vista previa de datos:

Elija el ícono de configuración (un símbolo de engranaje) para configurar sus preferencias para las previsualizaciones de datos. Esta configuración se aplica a todos los nodos del diagrama de trabajo. Puede hacer lo siguiente:

- Elija ajustar el texto de una línea a la siguiente. Esta opción está habilitada de forma predeterminada
- Cambie el número de filas (200 por defecto)
- Elija a un rol de IAM o cree un rol de IAM si es necesario
- Elija iniciar automáticamente una nueva sesión al crear un trabajo. Esto aprovisiona una nueva sesión interactiva al crear trabajos. Esta configuración se aplica en el nivel de la cuenta. Una vez configurada, se aplicará a todos los usuarios de su cuenta al editar cualquier trabajo.
- Elija la opción de inferir automáticamente el esquema. Se deducirán automáticamente los esquemas de salida para el nodo seleccionado
- Elija importar bibliotecas de AWS Glue automáticamente. Esto resulta útil, ya que evitará que la vista previa de los datos reinicie nuevas sesiones al añadir nuevas transformaciones que requieran el reinicio de la sesión

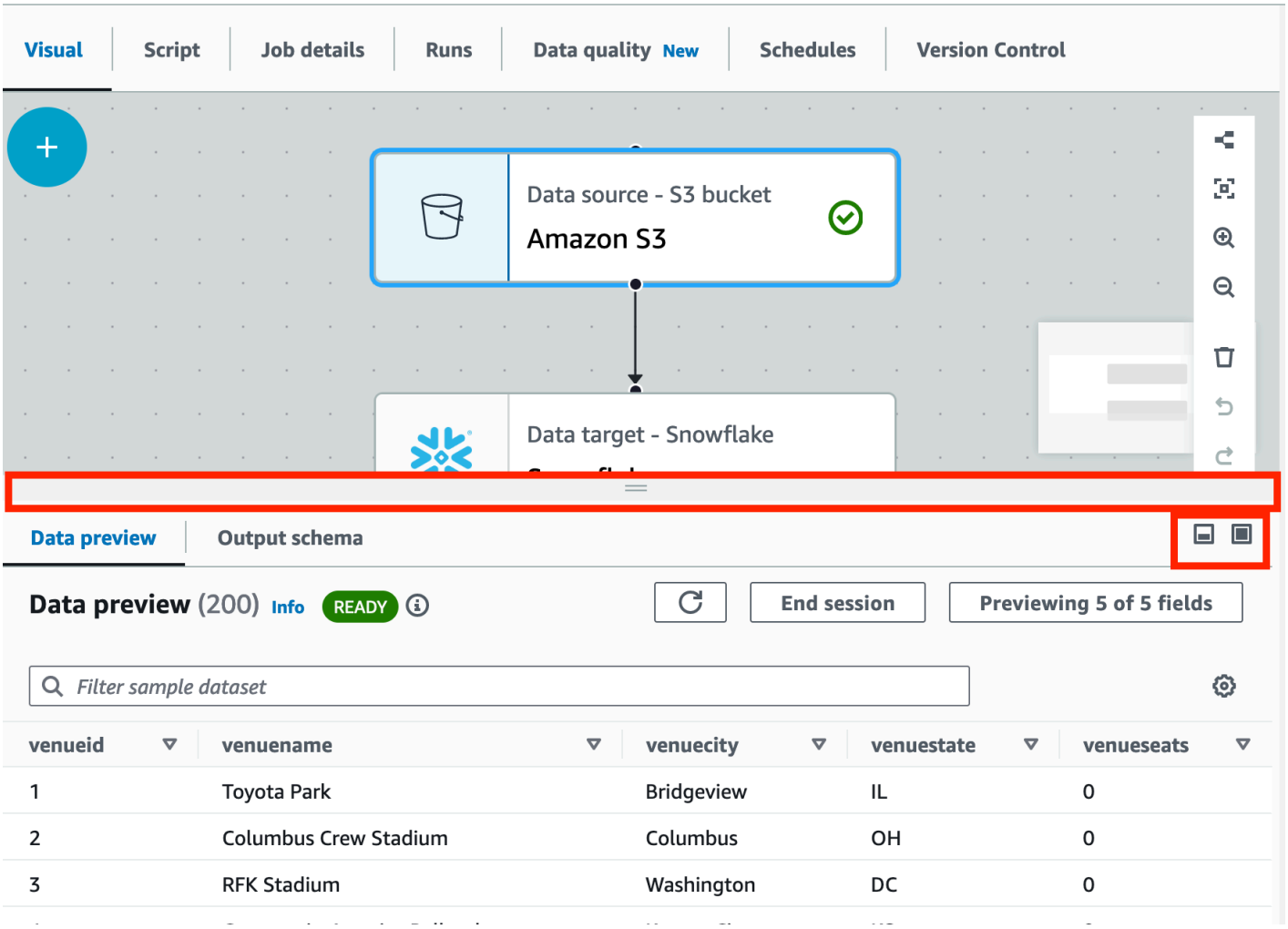


Las funciones adicionales incluyen la capacidad de:

- Elija el boton Previsualización de x de y campos para seleccionar las columnas (campos) que ds esea previsualizar. Al obtener una previsualización de los datos utilizando la configuración

predeterminada, el editor de trabajos muestra las primeras cinco columnas del conjunto de datos. Puede cambiar esto para mostrar todos o ninguno (no recomendado).

- Desplazarse por la ventana de previsualización de datos en forma horizontal como vertical.
- Utilice el botón de maximizar para expandir la pestaña de Previsualización de datos y superponerla sobre el gráfico del trabajo para ver mejor los datos y las estructuras de datos. Del mismo modo, utilice el botón de minimizar para minimizar la pestaña de previsualización de datos. También puede agarrar el panel de control y arrastrarlo hacia arriba para expandir la pestaña de Previsualización de datos.



- Utilice Finalizar sesión para detener la vista previa de los datos. Al detener la sesión, puede elegir un nuevo rol de IAM y establecer ajustes adicionales (como activar o desactivar los ajustes para iniciar automáticamente una nueva sesión, deducir un esquema o importar bibliotecas AWS Glue e iniciar la sesión de nuevo).

Restricciones al usar previsualizaciones de datos

Si utiliza las previsualizaciones de datos, podría encontrarse con las siguientes restricciones o limitaciones.

- La primera vez que elija la pestaña Previsualización de datos, deberá elegir un rol de IAM. Este rol debe tener los permisos correspondientes para acceder a los datos y otros recursos necesarios para crear las previsualizaciones de datos.
- Después de proporcionar un rol de IAM, tarda un tiempo antes de que los datos estén disponibles para su visualización. Para conjuntos de datos con menos de 1 GB de datos, puede tardar hasta un minuto. Si tiene un conjunto de datos grande, debe usar particiones para mejorar el tiempo de carga. La carga de datos directamente desde Amazon S3 ofrece el mejor rendimiento.
- Si tiene un conjunto de datos muy grande y tarda más de 15 minutos en consultar los datos para la previsualización de datos, se agotará el tiempo de espera de la solicitud. Las vistas previas de datos tienen un tiempo de espera de inactividad de 30 minutos. Para aliviar este problema, reduzca el tamaño del conjunto de datos que previsualizará.
- De forma predeterminada, verá las primeras 50 columnas en la pestaña Previsualización de datos. Si las columnas no tienen valores de datos, recibirá un mensaje que indicará que no hay datos para mostrar. Puede aumentar el número de filas muestreadas o seleccionar columnas diferentes para ver los valores de los datos.
- Actualmente, las previsualizaciones de datos no se soportan para orígenes de datos de streaming ni para orígenes de datos que utilizan conectores personalizados.
- Los errores en un nodo afectan a todo el trabajo. Si un nodo tiene un error con las previsualizaciones de datos, el error aparecerá en todos los nodos hasta que lo corrija.
- Si cambia un origen de datos para el trabajo, es posible que sea necesario actualizar los nodos secundarios de ese origen de datos para que coincidan con el nuevo esquema. Por ejemplo, si tiene un nodo ApplyMapping que modifica una columna y la columna no existe en el origen de datos de reemplazo, deberá actualizar el nodo de transformación ApplyMapping.
- Si ve la pestaña Previsualización de datos de un nodo de transformación de consulta SQL y la consulta SQL utiliza un nombre de campo incorrecto, la pestaña Previsualización de datos muestra un error.

Generación de código de script

Cuando se utiliza el editor visual para crear un trabajo, el código ETL se genera de forma automática. AWS Glue Studio crea un script de trabajo funcional y completo y lo guarda en una ubicación de Amazon S3.

Hay dos formas de código generadas por AWS Glue Studio: la versión original o clásica y una versión más nueva y optimizada. De forma predeterminada, el nuevo generador de código se utiliza para crear el script de trabajo. Puede generar un script de trabajo mediante el generador de código clásico en la pestaña Script al seleccionar el botón de alternar Generar script clásico.

Algunas de las diferencias en la nueva versión del código generado incluyen:

- Ya no se agregan bloques de comentarios grandes al script
- Las estructuras de salida del código utilizan el nombre de nodo que especifica en el editor visual. En el script de clase, las estructuras de salida se denominan tan solo `DataSource0`, `DataSource1`, `Transform0`, `Transform1`, `DataSink0`, `DataSink1` y así sucesivamente.
- Los comandos largos se dividen en varias líneas para eliminar la necesidad de desplazarse por la página a fin de ver todo el comando.

Nuevas características en AWS Glue Studio requieren la nueva versión de generación de código y no funcionará con el script de código clásico. Se le pide que actualice estos trabajos cuando intente ejecutarlos.

Edición de nodos de transformación de datos administrados por AWS Glue

AWS Glue Studio ofrece dos tipos de transformaciones:

- Transformaciones nativas de AWS Glue: disponibles para todos los usuarios y administradas por AWS Glue.
- Transformaciones visuales personalizadas: le permiten cargar sus propias transformaciones para usarlas en AWS Glue Studio

Nodos de transformación de datos administrados por AWS Glue

AWS Glue Studio proporciona un conjunto de transformaciones integradas que puede utilizar para procesar los datos. Sus datos pasan de un nodo en el diagrama de trabajo a otro en una estructura

de datos denominada `DynamicFrame`, que es una extensión de un `DataFrame` de Apache Spark SQL.

En el diagrama rellenado previamente para un trabajo, entre el origen de datos y los nodos de destino de datos se encuentra el nodo de transformación Cambiar esquema. Puede configurar este nodo de transformación para modificar los datos o puede utilizar transformaciones adicionales.

AWS Glue Studio dispone de las siguientes transformaciones integradas:

- ChangeSchema: asigne claves de propiedad de datos en el origen de datos a claves de propiedad de datos en el destino de datos. Puede cambiar el nombre de las claves, modificar los tipos de datos de las claves y elegir las claves que desea descartar del conjunto de datos.
- SelectFields (Seleccionar campos): elija las claves de propiedad de datos que desee conservar.
- DropFields (Descartar campos): elija las claves de propiedad de datos que desee descartar.
- RenameField (Renombrar campos): cambie el nombre de una sola clave de propiedad de datos.
- Spigot: escriba muestras de los datos en un bucket de Amazon S3.
- Join (Combinar): combine dos conjuntos de datos en uno mediante una frase de comparación en las claves de propiedad de datos especificadas. Puede utilizar combinaciones interna, externa, izquierda, derecha, semicombinación izquierda y anticombinación izquierda.
- Unión: combine filas de más de un origen de datos que tengan el mismo esquema.
- SplitFields (Dividir campos): divida claves de propiedad de datos en dos `DynamicFrames`. La salida es una recopilación de `DynamicFrames`: uno con las claves de propiedad de datos seleccionadas y el otro con las claves de propiedad de datos restantes.
- SelectFromCollection (Seleccionar desde recopilación): elija un `DynamicFrame` de una recopilación de `DynamicFrames`. La salida es el `DynamicFrame` seleccionado.
- FillMissingValues (Completar valores faltantes): para localizar registros en el conjunto de datos que tienen valores faltantes y agregar un nuevo campo con un valor sugerido determinado por imputación
- Filter (Filtro): divida un conjunto de datos en dos, en función de una condición de filtro.
- Eliminar campos nulos: elimina columnas del conjunto de datos si todos los valores de la columna son ‘nulos’.
- Eliminar duplicados: elimina las filas del origen de datos. Para ello, elija hacer coincidir filas enteras o especificar claves.
- SQL: ingrese el código SparkSQL en un campo de entrada de texto para utilizar una consulta SQL a fin de transformar los datos. La salida es un único `DynamicFrame`.

- Agregación: realiza un cálculo (como el promedio, la suma, el mínimo o el máximo) en los campos y filas seleccionados y crea un nuevo campo con los valores calculados recientes.
- Aplanar: extrae los campos dentro de las estructuras en los campos de nivel superior.
- UUID: agrega una columna con un Identificador único universal (UUID) para cada fila.
- Identificador: agrega una columna con un identificador numérico para cada fila.
- Para agregar una marca temporal: convierte una columna al tipo de marca temporal.
- Formato de marca temporal: convierte una columna de marca temporal en una cadena formateada.
- Transformación de Enrutador condicional: aplique múltiples condiciones a los datos entrantes. Cada fila de los datos entrantes se evalúa mediante una condición de filtro de grupo y se procesa en su grupo correspondiente.
- Transformación de concatenar columnas: cree una nueva columna de cadena. Para ello, utilice los valores de otras columnas con un espaciador opcional.
- Transformación de cadena dividida: divida una cadena en una matriz de tokens. Para ello, utilice una expresión regular para definir cómo se realiza la división.
- Transformación de matriz a columnas: extraiga algunos o todos los elementos de una columna de tipo matriz en nuevas columnas.
- Agregar la transformación de marca de tiempo actual: marque las filas con la hora a la que se procesaron los datos. Esto resulta útil para fines de auditoría o para realizar un seguimiento de la latencia en la canalización de datos.
- Transformación de filas dinámicas en columnas: agregue una columna numérica. Para ello, rote los valores únicos en las columnas seleccionadas para convertirlas en columnas nuevas. Si se seleccionan varias columnas, los valores se concatenan para dar nombre a las nuevas columnas.
- Transformación de columnas a filas: convierta columnas en valores de nuevas columnas. Para ello, genere una fila para cada valor único.
- Transformación de procesamiento de equilibrio automático: redistribuya mejor los datos entre los trabajadores. Esto es útil cuando los datos están desequilibrados o cuando provienen del origen y no permiten un procesamiento paralelo suficiente.
- Transformación de columnas derivadas: defina una nueva columna basándose en una fórmula matemática o expresión SQL en la que pueda utilizar otras columnas de los datos, así como constantes y literales.
- Transformación de búsqueda: agregue columnas de una tabla de catálogo definida cuando las claves coincidan con las columnas de búsqueda definidas en los datos.

- Transformación de matriz o mapa en filas: extraiga valores de una estructura anidada en filas individuales que sean más fáciles de manipular.
- Transformación de coincidencia de registros: invoque una transformación de clasificación de datos de machine learning de Record Matching existente.
- Transformación de eliminar filas nulas: elimine del conjunto de datos las filas que tengan todas las columnas nulas o vacías.
- Transformación de analizar columnas JSON: analice una columna de cadena que contenga datos de JSON y conviértala en una columna de estructura o matriz, en función de si el JSON es un objeto o una matriz, respectivamente.
- Transformación de extraer la ruta JSON: extraiga nuevas columnas de una columna de cadena JSON.
- Extraer fragmentos de cadenas de una expresión regular: extraiga fragmentos de cadenas mediante una expresión regular y cree una nueva columna a partir de ella o varias columnas si utiliza grupos de expresiones regulares.
- Custom transform (Transformación personalizada): ingrese un código en un campo de entrada de texto para utilizar transformaciones personalizadas. La salida es una recopilación de `DynamicFrames`.

Uso de una receta de preparación de datos en AWS Glue Studio

AWS Glue Studio permite utilizar una receta de AWS Glue DataBrew en un flujo de trabajo visual. Esto permite que las recetas de un cliente de AWS Glue DataBrew se ejecuten en un trabajo de AWS Glue junto con otros nodos de AWS Glue Studio.

En DataBrew, una receta es un conjunto de pasos de transformación de datos. Las recetas de DataBrew indican cómo transformar los datos que ya se han leído y no describen dónde y cómo leer datos, ni cómo y dónde escribirlos. Esto se configura en los nodos de origen y destino en AWS Glue Studio. Para obtener más información sobre las recetas, consulte Creación y uso de recetas de AWS Glue DataBrew.

El nodo Receta de preparación de datos está disponible en el panel de Recursos. Puede conectar el nodo Receta de preparación de datos a otro nodo del flujo de trabajo visual, ya sea un nodo de origen de datos u otro nodo de transformación. Tras elegir una receta de AWS Glue DataBrew y una versión, los pasos aplicados en la receta aparecen visibles en la pestaña de propiedades del nodo.

Requisitos previos

- Tiene una receta de AWS Glue DataBrew creada en AWS Glue DataBrew.
- Tiene los permisos de IAM necesarios como se describe en la siguiente sección.

Permisos de IAM para AWS Glue DataBrew

En este tema se proporciona información para ayudarle a comprender las acciones y los recursos que un administrador de IAM puede utilizar en una política de AWS Identity and Access Management (IAM) para la transformación de la receta de preparación de datos.

Para obtener información adicional sobre la seguridad en AWS Glue, consulte Gestión de acceso.

En la siguiente tabla, se enumeran los permisos que necesita un usuario para poder llevar a cabo operaciones específicas con el fin de usar la transformación de la receta de preparación de datos.

Receta de preparación de datos: acciones de transformación

Acción	Descripción
`databrew:ListRecipes`	Concede permiso para recuperar las recetas de AWS Glue DataBrew.
`databrew:ListRecipeVersions`	Concede permiso para recuperar las versiones de una receta de AWS Glue DataBrew.
`databrew:DescribeRecipe`	Concede permiso para recuperar la descripción de una receta de AWS Glue DataBrew.

El rol que utiliza para acceder a esta funcionalidad debe tener una política que permita varios AWS Glue DataBrew. Puede lograr esto al utilizar la política de `AWSGlueConsoleFullAccess` que incluye las acciones necesarias o al agregar la siguiente política interna al rol:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": [
                "databrew:ListRecipes",
                "databrew:ListRecipeVersions",
                "databrew:DescribeRecipe"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Para utilizar la transformación de la receta de preparación de datos, debe agregar la acción de `IAM:PassRole` a la política de permisos.

Permisos necesarios adicionales

Acción	Descripción
`iam:PassRole`	Otorga permiso a IAM para que el usuario pueda pasar las funciones aprobadas.

Sin estos permisos, se produce el siguiente error:

```
"errorCode": "AccessDenied"
"errorMessage": "User: arn:aws:sts::account_id:assumed-role/AWSGlueServiceRole is not
authorized to perform: iam:PassRole on resource: arn:aws:iam::account_id:role/service-
role/AWSGlueServiceRole
because no identity-based policy allows the iam:PassRole action"
```

Limitaciones

- No todas las recetas de AWS Glue DataBrew son compatibles con AWS Glue. Algunas recetas no se podrán ejecutar en AWS Glue Studio.
  - No se admiten recetas con transformaciones UNION y JOIN. Sin embargo, AWS Glue Studio ya cuenta con los nodos de transformación “Join” y “Union”, que se pueden utilizar antes o después de un nodo de recetas de preparación de datos.

- Los nodos de Receta de preparación de datos son compatibles con los trabajos que comiencen con la versión 4.0 de AWS Glue. Esta versión se seleccionará automáticamente después de agregar un nodo Receta de preparación de datos al trabajo.
- Los nodos de Receta de preparación de datos requieren Python. Esto se establece automáticamente cuando se agrega el nodo de Receta de preparación de datos al trabajo.
- Al utilizar la vista previa de datos, tendrá que reiniciar la sesión de vista previa de datos, después de agregar un nodo de Receta de preparación de datos al trabajo.

Cómo usar recetas de AWS Glue DataBrew en AWS Glue Studio

Para usar recetas de AWS Glue DataBrew en AWS Glue Studio, comience por crear recetas en AWS Glue DataBrew. Puede omitir este paso si ya dispone de las recetas que desea utilizar.

Para crear una nueva receta de AWS Glue DataBrew en AWS Glue DataBrew:

1. Cree una receta en AWS Glue DataBrew. Para obtener más información, consulte Introducción a AWS Glue DataBrew.
2. Guarde la receta.
3. Publique la receta. De este modo, la receta se publicará en la versión 1.0.

Para usar un nodo de receta de preparación de datos en AWS Glue Studio:

Puede usar más de un nodo de receta de preparación de datos en un trabajo de ETL visual. Para agregar un nodo de receta de preparación de datos siga los pasos que se indican a continuación y agregue otro nodo de receta de preparación de datos al trabajo. Por ejemplo, un flujo de trabajo puede seguir este patrón:

- Origen de datos 1 > receta 1 > salida 1
- Origen de datos 2 > receta 2 > salida 2
- salida 1, salida 2 > JOIN

1. Inicie un trabajo de AWS Glue en AWS Glue Studio con un origen de datos.
2. Agregue el nodo de Receta de preparación de datos al origen de datos.
3. Filtre la receta por nombre al escribirlo en el campo de búsqueda.
4. Elija la versión publicada. Solo están disponibles las versiones publicadas.

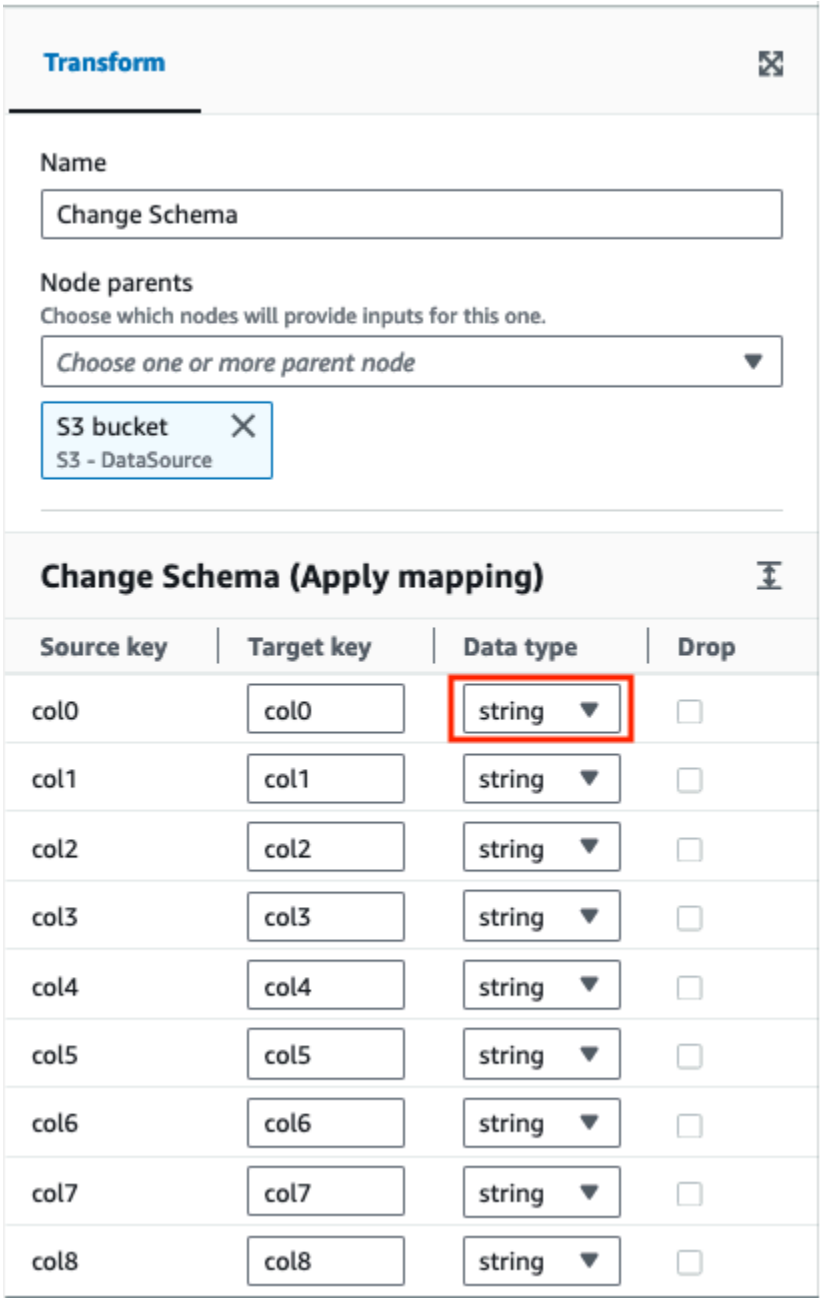
5. Para terminar de crear el trabajo, agregue otros nodos de transformación según sea necesario y agregue los nodos de destino de datos para guardar el resultado del trabajo.
6. Realice los cambios de configuración necesarios en la pestaña Detalles del trabajo, como asignarle un nombre al trabajo y ajustar la capacidad asignada según sea necesario y guarde el trabajo.
7. Ejecute el trabajo al seleccionar Ejecutar en el menú desplegable Acciones.

Para cambiar el esquema si el origen de datos es Amazon S3 y el formato de datos es CSV:

Si todas las columnas de un archivo CSV se cargan inicialmente como cadena de datos en AWS Glue Studio, debe asegurarse de que el tipo de datos de la columna sea compatible con el resto de los pasos de la receta de AWS Glue DataBrew.

Las recetas de AWS Glue DataBrew solo indican cómo transformar los datos que ya se han leído. No describen dónde y cómo leer los datos.

1. Agregue un nodo de Cambiar esquema antes del nodo de receta de varios pasos.
2. Haga clic en el nodo Cambiar esquema y cambie el esquema para que sea igual al de los tipos de datos de las columnas. Para ello, seleccione el nuevo tipo de datos en la Transformación de columnas en AWS Glue DataBrew, según sea necesario.



Para cambiar el esquema si el origen de datos no tiene encabezados:

Las recetas de AWS Glue DataBrew solo indican cómo transformar los datos que ya se han leído. No describen dónde y cómo leer los datos.

Al cargar conjuntos de datos sin encabezado en AWS Glue Studio, los nombres de encabezado predeterminados son diferentes de los que están cargados en AWS Glue DataBrew.

1. En el trabajo de ETL, agregue un nodo de Cambio de esquema antes del nodo de Receta de preparación de datos.
2. Elija el nodo Cambio de esquema y cambie los nombres de las columnas por los mismos nombres utilizados en la receta AWS Glue DataBrew.

Uso de Cambiar esquema para reasignar claves de propiedades de datos

Una transformación Cambiar esquema vuelve a mapear las claves de propiedad de datos de origen en la configuración deseada para los datos de destino. En un nodo de transformación de Cambiar esquema, puede:

- Cambiar el nombre de varias claves de propiedad de datos.
- Cambiar el tipo de datos de las claves de propiedad de datos, si se soporta el nuevo tipo de datos y existe una ruta de transformación entre los dos tipos de datos.
- Elegir un subconjunto de claves de propiedad de datos al indicar las claves de propiedad de datos que desea descartar.

También puede agregar nodos de esquema de cambio adicionales al diagrama de trabajo según sea necesario, por ejemplo, para modificar fuentes de datos adicionales o después de una transformación Join.

Uso de Change Schema con un tipo de datos decimal

Cuando se utiliza la transformación Change Schema con un tipo de datos decimal, la transformación Change Schema modifica la precisión al valor predeterminado de (10,2). Para modificarlo y establecer la precisión para su caso de uso, puede usar la transformación de consulta SQL y moldear las columnas con una precisión específica.

Por ejemplo, si tiene una columna de entrada llamada «DecimalCol» de tipo Decimal y desea reasignarla a una columna de salida llamada «OutputDecimalCol» con una precisión específica de (18,6), debería:

1. Añada una transformación de consulta SQL posterior después de la transformación de cambio de esquema.
2. En la transformación de consulta SQL, utilice una consulta SQL para convertir la columna reasignada con la precisión deseada. La consulta SQL tendría este aspecto:

```
SELECT col1, col2, CAST(DecimalCol AS DECIMAL(18,6)) AS OutputDecimalCol
```

```
FROM __THIS__
```

En la consulta SQL anterior:

- `col1` y `col2` son otras columnas de sus datos que desea revisar sin modificarlas.
- `DecimalCol` es el nombre de la columna original de los datos de entrada.
- `CAST (DecimalCol AS DECIMAL (18,6)) `convierte el` DecimalCol `en un tipo decimal con una precisión de 18 dígitos y 6 decimales.
- `AS OutputDecimalCol `cambia el nombre de la columna convertida a` `. OutputDecimalCol

Al utilizar la transformación de consulta SQL, puede anular la precisión predeterminada establecida por la transformación de cambio de esquema y convertir explícitamente las columnas decimales con la precisión deseada. Este enfoque le permite aprovechar la transformación de esquema de cambio para cambiar el nombre y reestructurar los datos y, al mismo tiempo, gestionar los requisitos de precisión de las columnas decimales durante la posterior transformación de SQL Query.

Añadir una transformación de esquema de cambio a su trabajo

> **Note**
>
> La transformación Cambiar esquema no distingue mayúsculas de minúsculas.

Para agregar un nodo de transformación de Cambiar esquema al diagrama de trabajo

1. (Opcional) abra el panel de recursos y elija Change Schema para agregar una nueva transformación al diagrama de trabajo, si es necesario.
2. En la pestaña Propiedades del nodo, ingrese un nombre para el nodo en el diagrama de trabajo. Si todavía no está seleccionado un nodo principal, elija un nodo de la lista Node parents (Nodos principales) para utilizar como origen de entrada para la transformación.
3. Elija la pestaña Transformación en el panel de detalles del nodo.
4. Modifique el esquema de entrada:

   - Para cambiar el nombre de una clave de propiedad de datos, escriba el nuevo nombre de la clave en el campo Target key (Clave de destino).
   - Para cambiar el tipo de datos de una clave de propiedad de datos, elija el tipo de datos nuevo para la clave de la lista Data type (Tipo de datos).

- Para eliminar una clave de propiedad de datos del esquema de destino, seleccione la casilla Drop (Descartar) para esa clave.

5. (Opcional) después de configurar las propiedades del nodo de transformación, puede ver el esquema de datos para sus datos al seleccionar la pestaña Output schema (Esquema de salida) en el panel de detalles del nodo. La primera vez que elija esta pestaña para cualquier nodo de trabajo, se le pedirá que proporcione un rol de IAM para acceder a los datos. Si no ha especificado un rol de IAM en Job details (Detalles del trabajo), se le pedirá que ingrese un rol de IAM aquí.
6. (Opcional) después de configurar las propiedades del nodo y las propiedades de transformación, puede obtener una previsualización del conjunto de datos modificado si selecciona la opción Data preview (Previsualización de datos) en el panel de detalles del nodo. La primera vez que elija esta pestaña para cualquier nodo de trabajo, se le pedirá que proporcione un rol de IAM para acceder a los datos. Hay un costo asociado con el uso de este recurso y la facturación comienza tan pronto como proporcione un rol de IAM.

Uso de Eliminar duplicados

La transformación Eliminar duplicados elimina las filas del origen de datos ofreciéndole dos opciones. Puede eliminar la fila duplicada que sea completamente igual o elegir los campos que desee que coincidan y eliminar solo las filas en función de los campos que haya elegido.

Por ejemplo, en este conjunto de datos, tiene filas duplicadas en las que todos los valores de algunas filas son exactamente iguales a los de otra fila y algunos de los valores de las filas son iguales o diferentes.

Fila	Nombre	Correo electrónico	Edad	Estado	Nota
1	Alegría	alegría@gmail	33	NY	
2	Tim	tim@gmail	45	OH	
3	Rosa	rosa@gmail	23	NJ	
4	Tim	tim@gmail	42	OH	

Fila	Nombre	Correo electrónico	Edad	Estado	Nota
5	Rosa	rosa@gmail	23	NJ	
6	Tim	tim@gmail	42	OH	se trata de una fila duplicada y coincide completamente con todos los valores de la fila #4
7	Rosa	rosa@gmail	23	NJ	Se trata de una fila duplicada y coincide completamente con todos los valores de la fila #5

Si opta por hacer coincidir filas enteras, las filas 6 y 7 se eliminarán del conjunto de datos. El conjunto de datos ahora es el siguiente:

Fila	Nombre	Correo electrónico	Edad	Estado
1	Alegría	alegría@gmail	33	NY
2	Tim	tim@gmail	45	OH
3	Rosa	rosa@gmail	23	NJ

Fila	Nombre	Correo electrónico	Edad	Estado
4	Tim	tim@gmail	42	OH
5	Rosa	rosa@gmail	23	NJ

Si opta por especificar las claves, puede optar por eliminar las filas que coincidan con las palabras “nombre” y “correo electrónico”. Esto permite controlar mejor qué es una “fila duplicada” para el conjunto de datos. Al especificar “nombre” y “correo electrónico”, el conjunto de datos ahora es el siguiente:

Fila	Nombre	Correo electrónico	Edad	Estado
1	Alegría	alegría@gmail	33	NY
2	Tim	tim@gmail	45	OH
3	Rosa	rosa@gmail	23	NJ

Algunas cosas a tener en cuenta:

- Para que las filas se reconozcan como duplicadas, los valores distinguen entre mayúsculas y minúsculas. Todos los valores de las filas deben tener las mismas mayúsculas y minúsculas; esto se aplica a cualquier opción que elija (hacer coincidir filas enteras o Especificar claves).
- Todos los valores se leen como cadenas.
- La transformación Eliminar duplicados utiliza el comando dropDuplicates de Spark.
- Cuando se utiliza la transformación Eliminar duplicados, la primera fila se mantiene y las demás filas se eliminan.
- La transformación Eliminar duplicados no cambia el esquema del marco de datos. Si decide especificar claves, todos los campos se mantienen en el marco de datos resultante.

Uso de SelectFields (Seleccionar campos) para eliminar la mayoría de las claves de propiedad de datos

Puede crear un subconjunto de claves de propiedad de datos a partir del conjunto de datos mediante la transformación Seleccionar campos. Indique qué claves de propiedad de datos desea conservar y el resto se eliminan del conjunto de datos.

> **Note**
>
> La transformación Seleccionar campos distingue entre mayúsculas y minúsculas. Use ApplyMapping (Aplicar mapeo) si necesita una forma que no distinga entre mayúsculas y minúsculas para seleccionar campos.

Para agregar un nodo de transformación Seleccionar campos al diagrama de trabajo

1. (Opcional) abra el panel de recursos y elija SelectFields para agregar una nueva transformación al diagrama de trabajo, si es necesario.
2. En la pestaña Node properties (Propiedades del nodo), ingrese un nombre para el nodo en el diagrama de trabajo. Si todavía no se ha seleccionado un nodo principal, elija un nodo de la lista Node parents (Nodos principales) para utilizar como origen de entrada para la transformación.
3. Elija la pestaña Transform (Transformación) en el panel de detalles del nodo.
4. En el encabezado SelectFields (Seleccionar campos) elija las claves de propiedad de datos en el conjunto de datos que desea conservar. Las claves de propiedad de datos no seleccionadas se descartan del conjunto de datos.

   También puede seleccionar la casilla junto al encabezado de la columna Field (Campo) para elegir automáticamente todas las claves de propiedad de datos en el conjunto de datos. Luego, puede anular la selección de claves de propiedad de datos individuales para eliminarlas del conjunto de datos.
5. (Opcional) después de configurar las propiedades del nodo de transformación, puede ver el esquema de datos para sus datos al seleccionar la pestaña Output schema (Esquema de salida) en el panel de detalles del nodo. La primera vez que elija esta pestaña para cualquier nodo de trabajo, se le pedirá que proporcione un rol de IAM para acceder a los datos. Si no ha especificado un rol de IAM en Job details (Detalles del trabajo), se le pedirá que ingrese un rol de IAM aquí.

6. (Opcional) después de configurar las propiedades del nodo y las propiedades de transformación, puede obtener una previsualización del conjunto de datos modificado si selecciona la opción Data preview (Previsualización de datos) en el panel de detalles del nodo. La primera vez que elija esta pestaña para cualquier nodo de trabajo, se le pedirá que proporcione un rol de IAM para acceder a los datos. Hay un costo asociado con el uso de este recurso y la facturación comienza tan pronto como proporcione un rol de IAM.

Uso de DropFields (Descartar campos) para mantener la mayoría de las claves de propiedad

Puede crear un subconjunto de claves de propiedad de datos a partir del conjunto de datos mediante la transformación Descartar campos. Indique qué claves de propiedad de datos desea eliminar del conjunto de datos y el resto de las claves se conservan.

> **Note**
>
> La transformación Descartar campos distingue entre mayúsculas y minúsculas. Use Cambiar esquema si necesita una forma insensible a mayúsculas y minúsculas para seleccionar campos.

Para agregar un nodo de transformación Descartar campos al diagrama de trabajo

1. (Opcional) abra el panel de recursos y elija DropFields para agregar una nueva transformación al diagrama de trabajo, si es necesario.
2. En la pestaña Node properties (Propiedades del nodo), ingrese un nombre para el nodo en el diagrama de trabajo. Si todavía no se ha seleccionado un nodo principal, elija un nodo de la lista Node parents (Nodos principales) para utilizar como origen de entrada para la transformación.
3. Elija la pestaña Transform (Transformación) en el panel de detalles del nodo.
4. En el encabezado DropFields (Descartar campos), elija las claves de propiedad de datos que desea descartar del origen de datos.

   También puede seleccionar la casilla junto al encabezado de la columna Field (Campo) para elegir automáticamente todas las claves de propiedad de datos en el conjunto de datos. Luego, puede anular la selección de claves de propiedad de datos individuales para que se conserven en el conjunto de datos.

5. (Opcional) después de configurar las propiedades del nodo de transformación, puede ver el esquema de datos para sus datos al seleccionar la pestaña Output schema (Esquema de salida) en el panel de detalles del nodo. La primera vez que elija esta pestaña para cualquier nodo de trabajo, se le pedirá que proporcione un rol de IAM para acceder a los datos. Si no ha especificado un rol de IAM en Job details (Detalles del trabajo), se le pedirá que ingrese un rol de IAM aquí.
6. (Opcional) después de configurar las propiedades del nodo y las propiedades de transformación, puede obtener una previsualización del conjunto de datos modificado si selecciona la opción Data preview (Previsualización de datos) en el panel de detalles del nodo. La primera vez que elija esta pestaña para cualquier nodo de trabajo, se le pedirá que proporcione un rol de IAM para acceder a los datos. Hay un costo asociado con el uso de este recurso y la facturación comienza tan pronto como proporcione un rol de IAM.

Cambio de nombre de un campo en el conjunto de datos

Puede utilizar la transformación RenameField (Renombrar campo) para cambiar el nombre de una clave de propiedad individual en el conjunto de datos.

> **Note**
>
> La transformación Renombrar campo distingue entre mayúsculas y minúsculas. Use ApplyMapping (Aplicar mapeo) si necesita una transformación que no distinga entre mayúsculas y minúsculas.

> **Tip**
>
> Si utiliza la transformación Cambiar Esquema, puede cambiar el nombre de varias claves de propiedad de datos en el conjunto de datos con una única transformación.

Para agregar un nodo de transformación Renombrar campo al diagrama de trabajo

1. (Opcional) abra el panel de recursos y elija RenameField para agregar una nueva transformación al diagrama de trabajo, si es necesario.

2. En la pestaña Node properties (Propiedades del nodo), ingrese un nombre para el nodo en el diagrama de trabajo. Si todavía no se ha seleccionado un nodo principal, elija un nodo de la lista Node parents (Nodos principales) para utilizar como origen de entrada para la transformación.
3. Elija la pestaña Transform (Transformación).
4. En el encabezado Data field (Campo de datos), elija una clave de propiedad de los datos de origen y, a continuación, escriba un nuevo nombre en el cuadro New field name (Nombre de campo nuevo).
5. (Opcional) después de configurar las propiedades del nodo de transformación, puede ver el esquema de datos para sus datos al seleccionar la pestaña Output schema (Esquema de salida) en el panel de detalles del nodo. La primera vez que elija esta pestaña para cualquier nodo de trabajo, se le pedirá que proporcione un rol de IAM para acceder a los datos. Si no ha especificado un rol de IAM en Job details (Detalles del trabajo), se le pedirá que ingrese un rol de IAM aquí.
6. (Opcional) después de configurar las propiedades del nodo y las propiedades de transformación, puede obtener una previsualización del conjunto de datos modificado si selecciona la opción Data preview (Previsualización de datos) en el panel de detalles del nodo. La primera vez que elija esta pestaña para cualquier nodo de trabajo, se le pedirá que proporcione un rol de IAM para acceder a los datos. Hay un costo asociado con el uso de este recurso y la facturación comienza tan pronto como proporcione un rol de IAM.

Uso de Spigot para tomar muestras del conjunto de datos

Para probar las transformaciones realizadas por el trabajo, es posible que desee obtener una muestra de los datos para comprobar que la transformación funciona según lo previsto. La transformación Spigot escribe un subconjunto de registros del conjunto de datos en un archivo JSON en un bucket de Amazon S3. El método de muestreo de datos puede ser un número especificado de registros del principio del archivo o un factor de probabilidad utilizado para elegir los registros.

Para agregar un nodo de transformación Spigot al diagrama de trabajo

1. (Opcional) abra el panel de recursos y elija Spigot para agregar una nueva transformación al diagrama de trabajo, si es necesario.
2. En la pestaña Node properties (Propiedades del nodo), ingrese un nombre para el nodo en el diagrama de trabajo. Si todavía no se ha seleccionado un nodo principal, elija un nodo de la lista Node parents (Nodos principales) para utilizar como origen de entrada para la transformación.
3. Elija la pestaña Transform (Transformación) en el panel de detalles del nodo.

4. Escriba una ruta de Amazon S3 o elija Browse S3 (Examinar S3) para elegir una ubicación en Amazon S3. Esta es la ubicación donde el trabajo escribe el archivo JSON que contiene la muestra de datos.
5. Ingrese la información para el método de muestreo. Puede especificar un valor para Number of records (Número de registros) que se escribirá desde el principio del conjunto de datos y un Probability threshold (Umbral de probabilidad) (que se ingresa como un valor decimal con un valor máximo de 1) para seleccionar cualquier registro determinado.

   Por ejemplo, para escribir los primeros 50 registros del conjunto de datos, debe establecer Number of records en 50 y Probability threshold en 1 (100 %).

Combinación de conjuntos de datos

La transformación Join (Combinación) le permite combinar dos conjuntos de datos en uno. Especifique los nombres de clave en el esquema de cada conjunto de datos que desea comparar. El `DynamicFrame` de salida contiene las filas donde las claves cumplen la condición de combinación. Las filas de cada conjunto de datos que cumplen con la condición de combinación se combinan en una sola fila en el `DynamicFrame` de salida que contiene todas las columnas encontradas en cualquiera de los conjuntos de datos.

Para agregar un nodo de transformación de combinación al diagrama de trabajo

1. Si sólo hay un origen de datos disponible, debe agregar un nuevo nodo de origen de datos al diagrama de trabajo.
2. Elija uno de los nodos de origen para la combinación. Abra el panel de recursos y, a continuación, elija Unir para agregar una nueva transformación al diagrama de trabajo.
3. En la pestaña Node properties (Propiedades del nodo), ingrese un nombre para el nodo en el diagrama de trabajo.
4. En la pestaña Node properties (Propiedades del nodo), en el encabezado Node parents (Nodos principales), agregue un nodo principal para que haya dos conjuntos de datos que proporcionen entradas para la combinación. El principal puede ser un nodo de origen de datos o un nodo de transformación.

> **Note**
>
> Una combinación sólo puede tener dos nodos principales.

5. Elija la pestaña Transform (Transformación).

   Si aparece un mensaje que indica que hay nombres de clave conflictivos, puede:

   - Seleccionar Resolve it (Resolver) para agregar automáticamente un nodo de transformación ApplyMapping (Aplicar mapeo) en el diagrama de trabajo. El nodo Aplicar mapeo agrega un prefijo a cualquier clave del conjunto de datos que tenga el mismo nombre que una clave del otro conjunto de datos. Por ejemplo, si utiliza el valor predeterminado de **`right`**, cualquier clave en el conjunto de datos derecho que tenga el mismo nombre que una clave en el conjunto de datos izquierdo cambiará de nombre a `(right)`*`key name`*.
   - Agregue manualmente un nodo de transformación con anterioridad en el diagrama de trabajo para eliminar o cambiar el nombre de las claves en conflicto.

6. Elija el tipo de combinación en la lista Join type (Tipo de combinación).

   - Inner join (Combinación interna): devuelve una fila con columnas de ambos conjuntos de datos para cada coincidencia basada en la condición de combinación. Las filas que no satisfacen la condición de combinación no se devuelven.
   - Left join (Combinación izquierda): todas las filas del conjunto de datos izquierdo y solo las filas del conjunto de datos derecho que satisfacen la condición de combinación.
   - Right join (Combinación derecha): todas las filas del conjunto de datos derecho y solo las filas del conjunto de datos izquierdo que satisfacen la condición de combinación.
   - Outer join (Combinación externa): todas las filas de ambos conjuntos de datos.
   - Left semi join (Semicombinación izquierda): todas las filas del conjunto de datos izquierdo que tienen una coincidencia en el conjunto de datos derecho en función de la condición de combinación.
   - Left anti join (Anticombinación izquierda): todas las filas del conjunto de datos izquierdo que no tienen una coincidencia en el conjunto de datos derecho en función de la condición de combinación.

7. En la pestaña Transform (Transformación), en el encabezado Join conditions (Condiciones de combinación), elija Add condition (Agregar condición). Elija una clave de propiedad de cada conjunto de datos para comparar. Las claves de propiedad en el lado izquierdo del operador de comparación se conocen como el conjunto de datos izquierdo y las claves de propiedad de la derecha se denominan conjunto de datos derecho.

   Para condiciones de combinación más complejas, puede agregar claves coincidentes adicionales al seleccionar Add condition (Agregar condición) más de una vez.

Si agrega una condición por accidente, puede elegir el ícono de eliminación
( )
para eliminarla.

8. (Opcional) después de configurar las propiedades del nodo de transformación, puede ver el esquema de datos para sus datos al seleccionar la pestaña Output schema (Esquema de salida) en el panel de detalles del nodo. La primera vez que elija esta pestaña para cualquier nodo de trabajo, se le pedirá que proporcione un rol de IAM para acceder a los datos. Si no ha especificado un rol de IAM en Job details (Detalles del trabajo), se le pedirá que ingrese un rol de IAM aquí.
9. (Opcional) después de configurar las propiedades del nodo y las propiedades de transformación, puede obtener una previsualización del conjunto de datos modificado si selecciona la opción Data preview (Previsualización de datos) en el panel de detalles del nodo. La primera vez que elija esta pestaña para cualquier nodo de trabajo, se le pedirá que proporcione un rol de IAM para acceder a los datos. Hay un costo asociado con el uso de este recurso y la facturación comienza tan pronto como proporcione un rol de IAM.

Para obtener un ejemplo del esquema de salida de combinación, considere una combinación entre dos conjuntos de datos con las siguientes claves de propiedad:

```
Left: {id, dept, hire_date, salary, employment_status}
Right: {id, first_name, last_name, hire_date, title}
```

La combinación está configurada para que coincida en las claves `id` y `hire_date` mediante el operador de comparación =.

Debido a que ambos conjuntos de datos contienen claves `id` y `hire_date`, debe elegir Resolve it (Resolver) para agregar automáticamente el prefijo **`right`** a las claves del conjunto de datos correcto.

Las claves en el esquema de salida serían:

```
{id, dept, hire_date, salary, employment_status,
(right)id, first_name, last_name, (right)hire_date, title}
```

Uso de Unión para combinar filas

El nodo de transformación Unión se utiliza cuando se desean combinar filas de más de un origen de datos que tienen el mismo esquema.

Existen dos tipos de transformaciones de Unión:

1. ALL: al aplicar ALL, la unión resultante no elimina filas duplicadas.
2. DISTINCT: al aplicar DISTINTO, la unión resultante elimina las filas duplicadas.

Uniones versus Combinaciones

Se usa Union para combinar filas. Se utiliza Combinar para combinar columnas.

Uso de la transformación Unión en el lienzo de Visual ETL

1. Agregue más de un origen de datos para realizar una transformación de unión. Para agregar un origen de datos, abra el panel de recursos y, a continuación, elija el origen de datos en la pestaña Fuentes. Antes de usar la transformación de Unión, debe asegurarse de que todos los orígenes de datos involucradas en la unión tengan el mismo esquema y estructura.
2. Si tiene al menos dos orígenes de datos que desee combinar mediante la transformación Unión, cree la transformación Unión al agregarla al lienzo. Abre el panel de recursos en el lienzo y busca “Unión”. También puedes elegir la pestaña Transformaciones del panel de recursos y desplazarte hacia abajo hasta encontrar la transformación de Unión y, a continuación, elegir Unión.
3. Seleccione el nodo Unión en el lienzo de trabajo. En la ventana de propiedades del nodo, elija los nodos principales para conectarse a la transformación de Unión.
4. AWS Glue comprueba la compatibilidad para garantizar que la transformación de Unión se pueda aplicar a todas los orígenes de datos. Si el esquema de los orígenes de datos es el mismo, se permitirá la operación. Si los orígenes de datos no tienen el mismo esquema, aparece un mensaje de error no válido: “Los esquemas de entrada de esta unión no son los mismos. Considere la posibilidad de utilizar ApplyMapping para que coincida con los esquemas”. Para solucionar este problema, selecciona utilizar ApplyMapping.
5. Seleccione el tipo de Unión.
    1. Todas: de forma predeterminada, se selecciona el tipo de unión Todas; esto resultará en filas duplicadas si las hay en la combinación de datos.
    2. Distinto: elija Distinto si desea eliminar las filas duplicadas de la combinación de datos resultante.

Uso de SplitFields (Dividir campos) para dividir un conjunto de datos en dos

La transformación SplitFields (Dividir campos) le permite elegir algunas de las claves de propiedad de datos en el conjunto de datos de entrada para ponerlas en un conjunto de datos y colocar las claves no seleccionadas en otro conjunto de datos independiente. La salida de esta transformación es una recopilación de `DynamicFrames`.

> **Note**
>
> Debe utilizar la transformación SelectFromCollection (Seleccionar desde la recopilación) para convertir la recopilación de `DynamicFrames` en un solo `DynamicFrame` antes de enviar la salida a una ubicación de destino.

La transformación Dividir campos distingue entre mayúsculas y minúsculas. Agregue una transformación ApplyMapping (Aplicar mapeo)como nodo principal si necesita nombres de clave de propiedad que no distingan entre mayúsculas y minúsculas.

Para agregar un nodo de transformación Dividir campos al diagrama de trabajo

1. (Opcional) abra el panel de recursos y elija SplitFields para agregar una nueva transformación al diagrama de trabajo, si es necesario.
2. En la pestaña Node properties (Propiedades del nodo), ingrese un nombre para el nodo en el diagrama de trabajo. Si todavía no se ha seleccionado un nodo principal, elija un nodo de la lista Node parents (Nodos principales) para utilizar como origen de entrada para la transformación.
3. Elija la pestaña Transform (Transformación).
4. Elija las claves de propiedad que desea poner en el primer conjunto de datos. Las claves que no elija se colocan en el segundo conjunto de datos.
5. (Opcional) después de configurar las propiedades del nodo de transformación, puede ver el esquema de datos para sus datos al seleccionar la pestaña Output schema (Esquema de salida) en el panel de detalles del nodo. La primera vez que elija esta pestaña para cualquier nodo de trabajo, se le pedirá que proporcione un rol de IAM para acceder a los datos. Si no ha especificado un rol de IAM en Job details (Detalles del trabajo), se le pedirá que ingrese un rol de IAM aquí.
6. (Opcional) después de configurar las propiedades del nodo y las propiedades de transformación, puede obtener una previsualización del conjunto de datos modificado si selecciona la opción Data preview (Previsualización de datos) en el panel de detalles del nodo. La primera vez que

elija esta pestaña para cualquier nodo de trabajo, se le pedirá que proporcione un rol de IAM para acceder a los datos. Hay un costo asociado con el uso de este recurso y la facturación comienza tan pronto como proporcione un rol de IAM.

7. Configure un nodo de transformación SelectFromCollection (Seleccionar desde la recopilación) para procesar los conjuntos de datos resultantes.

Información general de la transformación SelectFromCollection

Algunas transformaciones tienen múltiples conjuntos de datos como salida en lugar de un único conjunto de datos, por ejemplo, Dividir campos. La transformación Seleccionar desde recopilación selecciona un conjunto de datos (`DynamicFrame`) desde una recopilación de conjuntos de datos (una matriz de `DynamicFrames`). El resultado de la transformación es el `DynamicFrame` seleccionado.

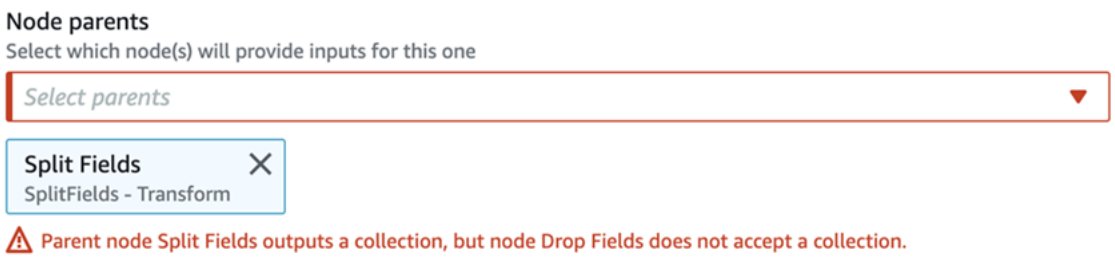
Debe utilizar esta transformación después de utilizar una transformación que crea una recopilación de `DynamicFrames`, por ejemplo:

- Transformaciones de código personalizado
- SplitFields

Si no agrega un nodo de transformación Seleccionar desde recopilación a su diagrama de trabajo después de cualquiera de estas transformaciones, su trabajo arrojará error.

El nodo principal de esta transformación debe ser un nodo que devuelve una recopilación de `DynamicFrames`. Si elige un nodo principal para este nodo de transformación que devuelve un único `DynamicFrame`, como una transformación Combinación, su trabajo arroja un error.

Del mismo modo, si usa un nodo Seleccionar desde recopilación en el diagrama de trabajo como nodo principal de una transformación que espera un único `DynamicFrame` como entrada, su trabajo arroja un error.



Uso de SelectFromCollection (Seleccionar desde la recopilación) para elegir qué conjunto de datos desea mantener

Use la transformación Seleccionar desde recopilación para convertir una recopilación de `DynamicFrames` en un solo `DynamicFrame`.

Para agregar un nodo de transformación Seleccionar desde recopilación al diagrama de trabajo

1. (Opcional) abra el panel de recursos y elija SelectFromCollection para agregar una nueva transformación al diagrama de trabajo, si es necesario.
2. En la pestaña Node properties (Propiedades del nodo), ingrese un nombre para el nodo en el diagrama de trabajo. Si todavía no se ha seleccionado un nodo principal, elija un nodo de la lista Node parents (Nodos principales) para utilizar como origen de entrada para la transformación.
3. Elija la pestaña Transform (Transformación).
4. En el encabezado Frame index (Índice del marco), elija el número de índice de la matriz que corresponde al `DynamicFrame` que desea seleccionar de la recopilación de `DynamicFrames`.

   Por ejemplo, si el nodo principal de esta transformación es Dividir campos, podrá ver el esquema para cada `DynamicFrame` en la pestaña Output schema (Esquema de salida) del nodo. Si desea mantener el `DynamicFrame` asociado al esquema para Output 2 (Salida 2), debería seleccionar **1** para el valor de Frame index (Índice del marco), que es el segundo valor de la lista.

   Solo el `DynamicFrame` que elija se incluye en la salida.
5. (Opcional) después de configurar las propiedades del nodo de transformación, puede ver el esquema de datos para sus datos al seleccionar la pestaña Output schema (Esquema de salida) en el panel de detalles del nodo. La primera vez que elija esta pestaña para cualquier nodo de trabajo, se le pedirá que proporcione un rol de IAM para acceder a los datos. Si no ha especificado un rol de IAM en Job details (Detalles del trabajo), se le pedirá que ingrese un rol de IAM aquí.
6. (Opcional) después de configurar las propiedades del nodo y las propiedades de transformación, puede obtener una previsualización del conjunto de datos modificado si selecciona la opción Data preview (Previsualización de datos) en el panel de detalles del nodo. La primera vez que elija esta pestaña para cualquier nodo de trabajo, se le pedirá que proporcione un rol de IAM para acceder a los datos. Hay un costo asociado con el uso de este recurso y la facturación comienza tan pronto como proporcione un rol de IAM.

Buscar y rellenar valores faltantes en un conjunto de datos

Puede utilizar la transformación FillMissingValues (Completar valores faltantes) para localizar registros en el conjunto de datos que tienen valores faltantes y agregar un nuevo campo con un valor determinado por imputación. El conjunto de datos de entrada se utiliza para brindar formación al modelo de machine learning (ML) que determina cuál debe ser el valor que falta. Si utiliza conjuntos de datos progresivos, cada conjunto progresivo se utiliza como datos de formación para el modelo de ML, por lo que es posible que los resultados no sean tan precisos.

Para utilizar un nodo de transformación FillMissingValues (Completar valores faltantes) en el diagrama de trabajo

1. (Opcional) abra el panel de recursos y elija FillMissingValues para agregar una nueva transformación al diagrama de trabajo, si es necesario.
2. En la pestaña Node properties (Propiedades del nodo), ingrese un nombre para el nodo en el diagrama de trabajo. Si todavía no está seleccionado un nodo principal, elija un nodo de la lista Node parents (Nodos principales) para utilizar como origen de entrada para la transformación.
3. Elija la pestaña Transform (Transformación).
4. Para Data field (Campo de datos), elija el nombre de columna o campo de los datos de origen que desea analizar para detectar los valores que faltan.
5. (Opcional) en el campo New field name (Nombre de campo nuevo), ingrese un nombre para el campo agregado a cada registro que contendrá el valor de reemplazo estimado para el campo analizado. Si el campo analizado no tiene un valor faltante, el valor del campo analizado se copia en el nuevo campo.

   Si no especifica un nombre para el campo nuevo, el nombre predeterminado es el nombre de la columna analizada con `_filled`asociado. Por ejemplo, si ingresa **`Age`** para el Data field (Campo de datos) y no especifica un valor para New field name (Nombre de campo nuevo), se agrega un nombre de campo nuevo **`Age_filled`** a cada registro.
6. (Opcional) después de configurar las propiedades del nodo de transformación, puede ver el esquema de datos para sus datos al seleccionar la pestaña Output schema (Esquema de salida) en el panel de detalles del nodo. La primera vez que elija esta pestaña para cualquier nodo de trabajo, se le pedirá que proporcione un rol de IAM para acceder a los datos. Si no ha especificado un rol de IAM en Job details (Detalles del trabajo), se le pedirá que ingrese un rol de IAM aquí.
7. (Opcional) después de configurar las propiedades del nodo y las propiedades de transformación, puede obtener una previsualización del conjunto de datos modificado si selecciona la opción

Data preview (Previsualización de datos) en el panel de detalles del nodo. La primera vez que elija esta pestaña para cualquier nodo de trabajo, se le pedirá que proporcione un rol de IAM para acceder a los datos. Hay un costo asociado con el uso de este recurso y la facturación comienza tan pronto como proporcione un rol de IAM.

Filtrado de claves dentro de un conjunto de datos

Use la transformación Filter (Filtro) para crear un nuevo conjunto de datos al filtrar los registros del conjunto de datos de entrada en función de una expresión regular. Las filas que no satisfacen la condición del filtro se eliminan de la salida.

- Para los tipos de datos de cadena, puede filtrar filas en las que el valor de clave coincida con una cadena especificada.
- Para los tipos de datos numéricos, puede filtrar filas mediante la comparación del valor clave con un valor especificado a través de los operadores de comparación <, >, =, !=, <= y >=.

Si especifica varias condiciones de filtro, los resultados se combinan mediante un operador AND de forma predeterminada, pero puede elegir OR en su lugar.

La transformación Filtro distingue entre mayúsculas y minúsculas. Agregue una transformación ApplyMapping (Aplicar mapeo)como nodo principal si necesita nombres de clave de propiedad que no distingan entre mayúsculas y minúsculas.

Para agregar un nodo de transformación de filtro al diagrama de trabajo

1. (Opcional) abra el panel de recursos y elija Filtro para agregar una nueva transformación al diagrama de trabajo, si es necesario.
2. En la pestaña Node properties (Propiedades del nodo), ingrese un nombre para el nodo en el diagrama de trabajo. Si todavía no se ha seleccionado un nodo principal, elija un nodo de la lista Node parents (Nodos principales) para utilizar como origen de entrada para la transformación.
3. Elija la pestaña Transform (Transformación).
4. Elija Global AND (Global Y) o Global OR (Global O). Esto determina cómo se combinan distintas condiciones de filtro. Todas las condiciones se combinan mediante operaciones AND o OR. Si solo tiene una sola condición de filtro, puede elegir entre cualquiera de las dos.
5. Elija el botón Add condition (Agregar condición) en la sección Filter condition (Condición de filtro) para agregar una condición de filtro.

En el campo Key (Clave), elija un nombre de clave de propiedad a partir del conjunto de datos. En el campo Operation (Operación) elija el operador de comparación. En el campo Value (Valor), ingrese el valor de comparación. Estas son algunas ejemplos de condiciones de filtro:

- `year >= 2018`
- `State matches 'CA*'`

Cuando filtra los valores de cadena, asegúrese de que el valor de comparación utiliza un formato de expresión regular que coincida con el lenguaje de script seleccionado en las propiedades del trabajo (Python o Scala).

6. Agregue condiciones de filtro adicionales, según sea necesario.
7. (Opcional) después de configurar las propiedades del nodo de transformación, puede ver el esquema de datos para sus datos al seleccionar la pestaña Output schema (Esquema de salida) en el panel de detalles del nodo. La primera vez que elija esta pestaña para cualquier nodo de trabajo, se le pedirá que proporcione un rol de IAM para acceder a los datos. Si no ha especificado un rol de IAM en Job details (Detalles del trabajo), se le pedirá que ingrese un rol de IAM aquí.
8. (Opcional) después de configurar las propiedades del nodo y las propiedades de transformación, puede obtener una previsualización del conjunto de datos modificado si selecciona la opción Data preview (Previsualización de datos) en el panel de detalles del nodo. La primera vez que elija esta pestaña para cualquier nodo de trabajo, se le pedirá que proporcione un rol de IAM para acceder a los datos. Hay un costo asociado con el uso de este recurso y la facturación comienza tan pronto como proporcione un rol de IAM.

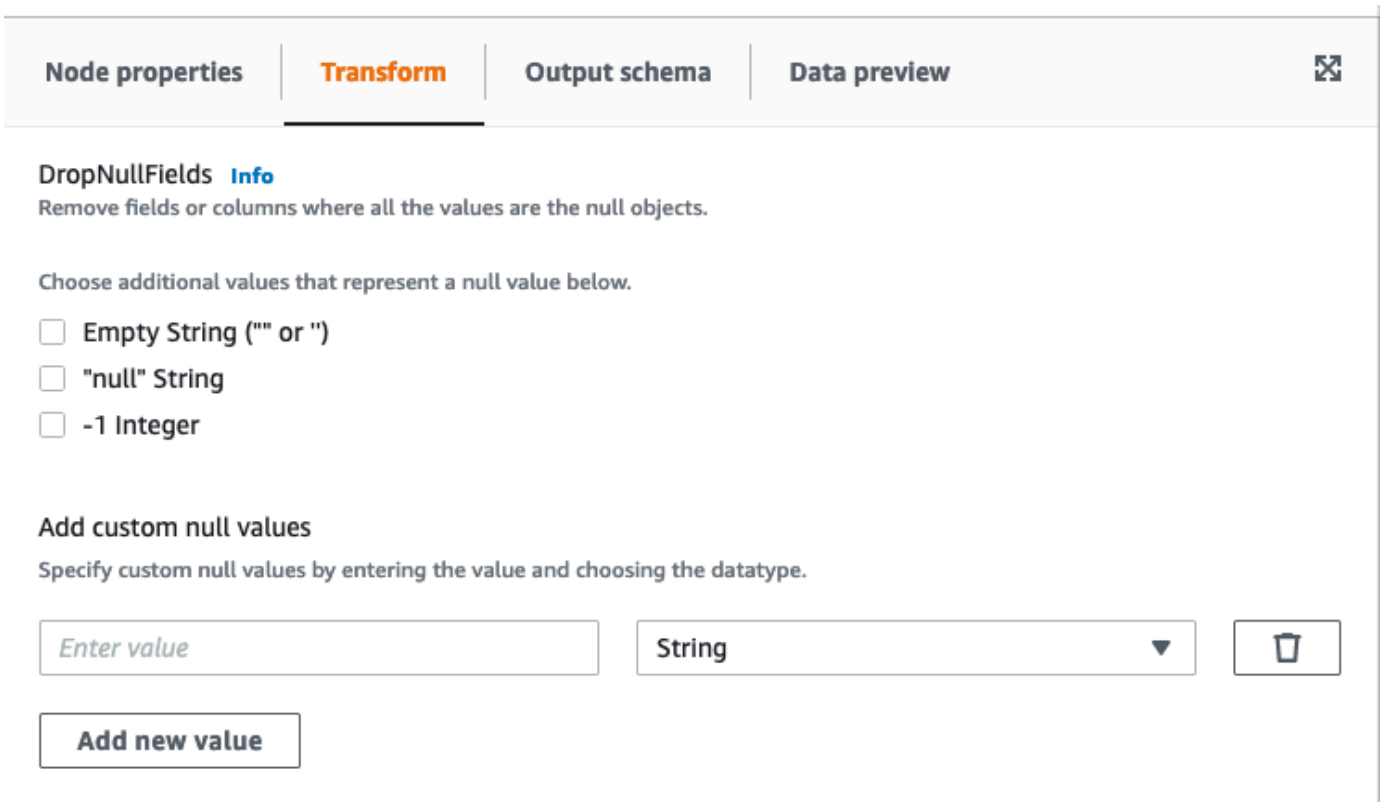
Uso de DropNullFields para eliminar campos con valores nulos

Utilice la transformación DropNullFields para eliminar campos del conjunto de datos si todos los valores son ‘nulos’. De forma predeterminada, AWS Glue Studio reconocerá objetos nulos, pero algunos valores, como cadenas vacías, cadenas que son “nulas”, los enteros -1 u otros marcadores de posición, como ceros, no se reconocen de manera automática como nulos.

Para utilizar DropNullFields

1. Agregue un nodo DropNullFields al diagrama de trabajo.

2. En la pestaña Node properties (Propiedades del nodo), elija valores adicionales que representen un valor nulo. Puede elegir seleccionar todos los valores o ninguno:



   - Cadena vacía (“” o “): los campos que contienen cadenas vacías se eliminarán
   - “cadena null”: los campos que contienen la cadena con la palabra ‘null’ se eliminarán
   - entero -1: los campos que contienen un entero -1 (uno negativo) se eliminarán
3. Si es necesario, también puede especificar valores nulos personalizados. Son valores nulos que pueden ser exclusivos de su conjunto de datos. Para agregar un valor nulo personalizado, elija Add new value (Agregar un nuevo valor).
4. Ingrese el valor nulo personalizado. Por ejemplo, puede ser cero o cualquier valor que se utilice para representar un valor nulo en el conjunto de datos.
5. Elija el tipo de datos en el campo desplegable. Los tipos de datos pueden ser String o Integer.

> **Note**
>
> Los valores nulos personalizados y los tipos de datos deben coincidir con exactitud para que los campos se reconozcan como valores nulos y se eliminen. Las coincidencias

parciales en las que solo coincide el valor nulo personalizado, pero el tipo de datos no lo hace, no provocarán que los campos se eliminen.

Uso de una consulta SQL para transformar datos

Puede usar una transformación SQL para escribir su propia transformación en forma de consulta SQL.

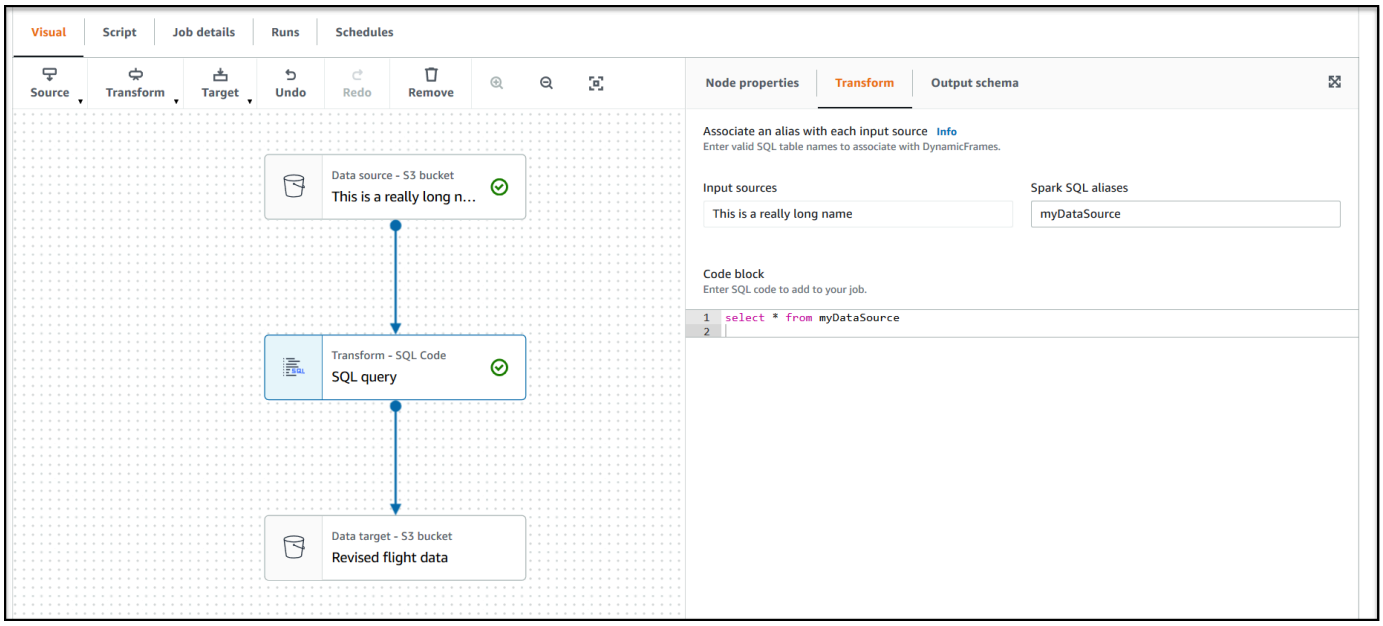
Un nodo de transformación SQL puede tener varios conjuntos de datos como entradas, pero solo produce un único conjunto de datos como salida. Contiene un campo de texto, donde se introduce la consulta de Apache SparkSQL. Puede asignar alias a cada conjunto de datos utilizado como entrada, para ayudar simplemente a la consulta SQL. Para obtener más información acerca de la sintaxis SQL, consulte documentación de Spark SQL.

> **Note**
>
> Si utiliza una transformación de Spark SQL con un origen de datos ubicado en una VPC, agregue un punto de enlace de desarrollo de la VPC de AWS Glue a la VPC que contenga el origen de datos. Para obtener más información acerca de la configuración de puntos de enlace de desarrollo, consulte Agregar un punto de enlace de desarrollo, Configuración del entorno para puntos de enlace de desarrollo y Acceso al punto de enlace de desarrollo en la Guía para desarrolladores de AWS Glue.

Para agregar un nodo de transformación SQL al diagrama de trabajo

1. (Opcional) agregue un nodo de transformación al diagrama de trabajo, de ser necesario. Seleccione Spark SQL para el tipo de nodo.
2. En la pestaña Node properties (Propiedades del nodo), ingrese un nombre para el nodo en el diagrama de trabajo. Si todavía no se ha seleccionado un nodo principal, o si desea varias entradas para la transformación SQL, elija un nodo en la lista Node parents (Nodos principales) que se utilizará como origen de entrada para la transformación. Agregue nodos principales adicionales según sea necesario.
3. Elija la pestaña Transform (Transformación) en el panel de detalles del nodo.
4. Los conjuntos de datos de origen para la consulta SQL se identifican mediante los nombres especificados en el campo Name (Nombre) para cada nodo. Si no desea utilizar estos nombres

o si los nombres no son adecuados para una consulta SQL, puede asociar un nombre a cada conjunto de datos. La consola proporciona alias predeterminados, como `MyDataSource`.



Por ejemplo, si un nodo principal para el nodo de transformación SQL se denomina `Rename Org PK field`, puede asociar el nombre `org_table` con este conjunto de datos. Este alias se puede utilizar en la consulta SQL en lugar del nombre del nodo.

5. En el campo de entrada de texto bajo el encabezado Code block (Bloque de código), pegue o escriba la consulta SQL. El campo de texto muestra resaltado de sintaxis SQL y sugerencias de palabras clave.
6. Con el nodo de transformación SQL seleccionado, elija la opción Output schema (Esquema de salida) y, a continuación, elija Edit (Editar). Proporcione las columnas y los tipos de datos que describen los campos de salida de la consulta SQL.

   Especifique el esquema mediante las siguientes acciones en la sección Output schema (Esquema de salida) de la página:

   - Para cambiar el nombre de una columna, coloque el cursor en el cuadro de texto Key (Clave) para la columna [también conocido como field (campo) o property key (clave de propiedad)] e ingrese el nuevo nombre.
   - Para cambiar el tipo de datos de una columna, seleccione el nuevo tipo de datos para la columna en la lista desplegable.
   - Para agregar una nueva columna de nivel superior al esquema, elija la opción Overflow (Desbordamiento)

(⋯)
y, a continuación, elija Add root key (Agregar clave raíz). Se agregan nuevas columnas en la parte superior del esquema.

- Para eliminar una columna del esquema, elija el ícono de eliminación
(🗑)
en el extremo derecho del nombre de la clave.

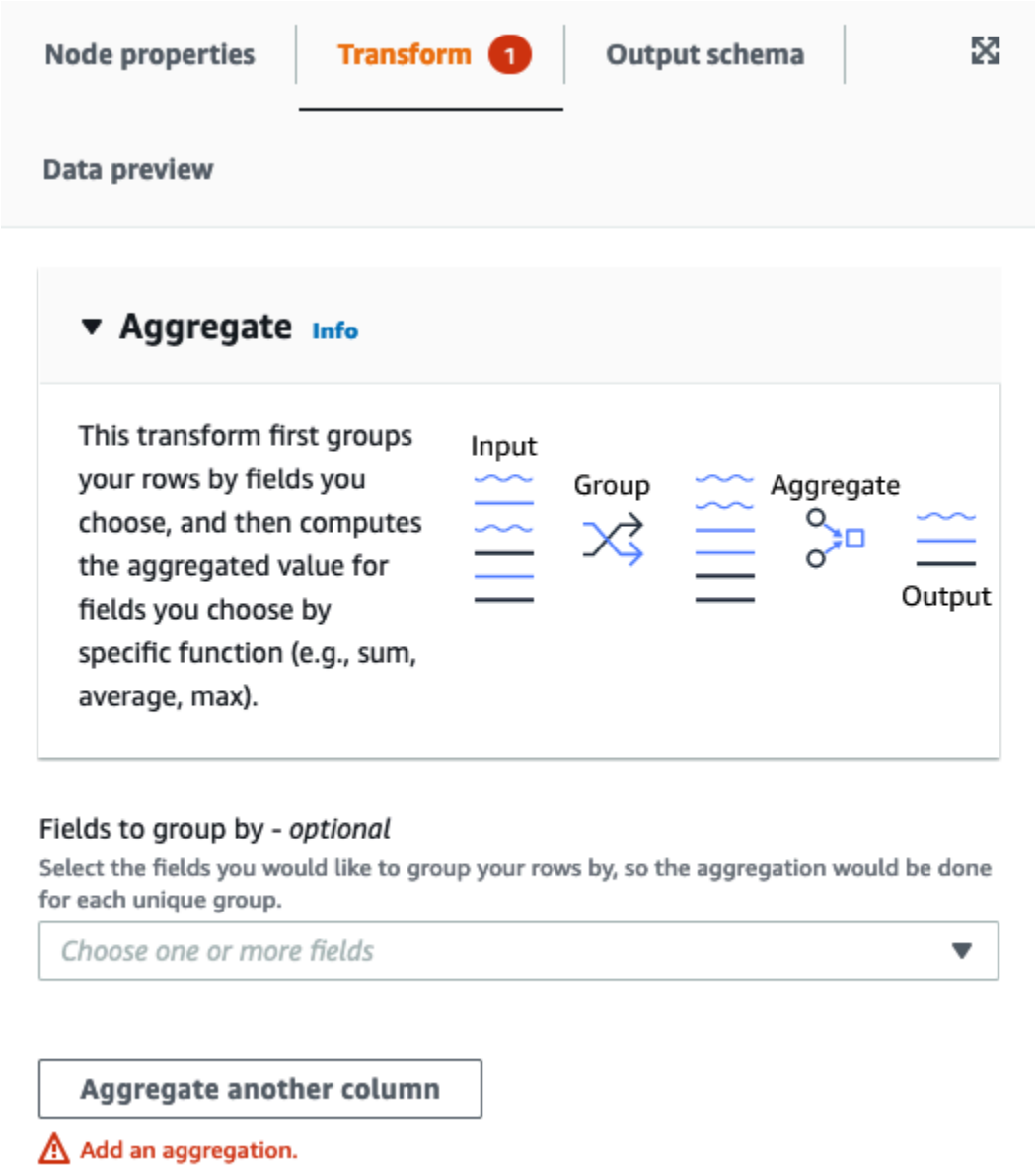
7. Cuando termine de especificar el esquema de salida, elija Apply (Aplicar) para guardar los cambios y salir del editor de esquemas. Si no desea guardar los cambios, elija Cancel (Cancelar) para editar el editor de esquemas.
8. (Opcional) después de configurar las propiedades del nodo y las propiedades de transformación, puede obtener una previsualización del conjunto de datos modificado si selecciona la opción Data preview (Previsualización de datos) en el panel de detalles del nodo. La primera vez que elija esta pestaña para cualquier nodo de trabajo, se le pedirá que proporcione un rol de IAM para acceder a los datos. Hay un costo asociado con el uso de este recurso y la facturación comienza tan pronto como proporcione un rol de IAM.

Uso de Agregar para realizar cálculos resumidos en campos seleccionados

Para utilizar la transformación agregada

1. Agregue el nodo agregado al diagrama de trabajos.
2. En la pestaña Node properties (Propiedades del nodo), elija los campos para agruparlos al seleccionar el campo desplegable (opcional). Puede seleccionar más de un campo a la vez o buscar un nombre de campo al escribir en la barra de búsqueda.

   Cuando se seleccionan los campos, se muestran el nombre y el tipo de datos. Para eliminar un campo, seleccione ‘X’.



3. Elija Aggregate another column (Agregar otra columna). Es necesario seleccionar al menos un campo.

4. Elija un campo en el Field to aggretate (Campo a agregar) desplegable.

5. Elija la función de agregación que desea aplicar al campo elegido:

   - avg: calcula el promedio
   - countDistinct: calcula el número de valores únicos no nulos
   - count: calcula el número de valores no nulos
   - first: devuelve el primer valor que satisface los criterios “agrupar por”
   - last: devuelve el último valor que satisface los criterios “agrupar por”
   - kurtosis: calcula la nitidez del pico de una curva de distribución de frecuencias
   - max: devuelve el valor más alto que satisface los criterios “agrupar por”
   - min: devuelve el valor más bajo que satisface los criterios “agrupar por”
   - sesgo: medida de la asimetría de la distribución de probabilidad de una distribución normal
   - stddev_pop: calcula la desviación estándar de la población y devuelve la raíz cuadrada de la variación de la población
   - sum: la suma de todos los valores en el grupo
   - sumDistinct: la suma de distintos valores en el grupo
   - var_samp: la variación de la muestra del grupo (ignora los valores nulos)
   - var_pop: la variación de la población del grupo (ignora los valores nulos)

Aplanamiento de estructuras anidadas

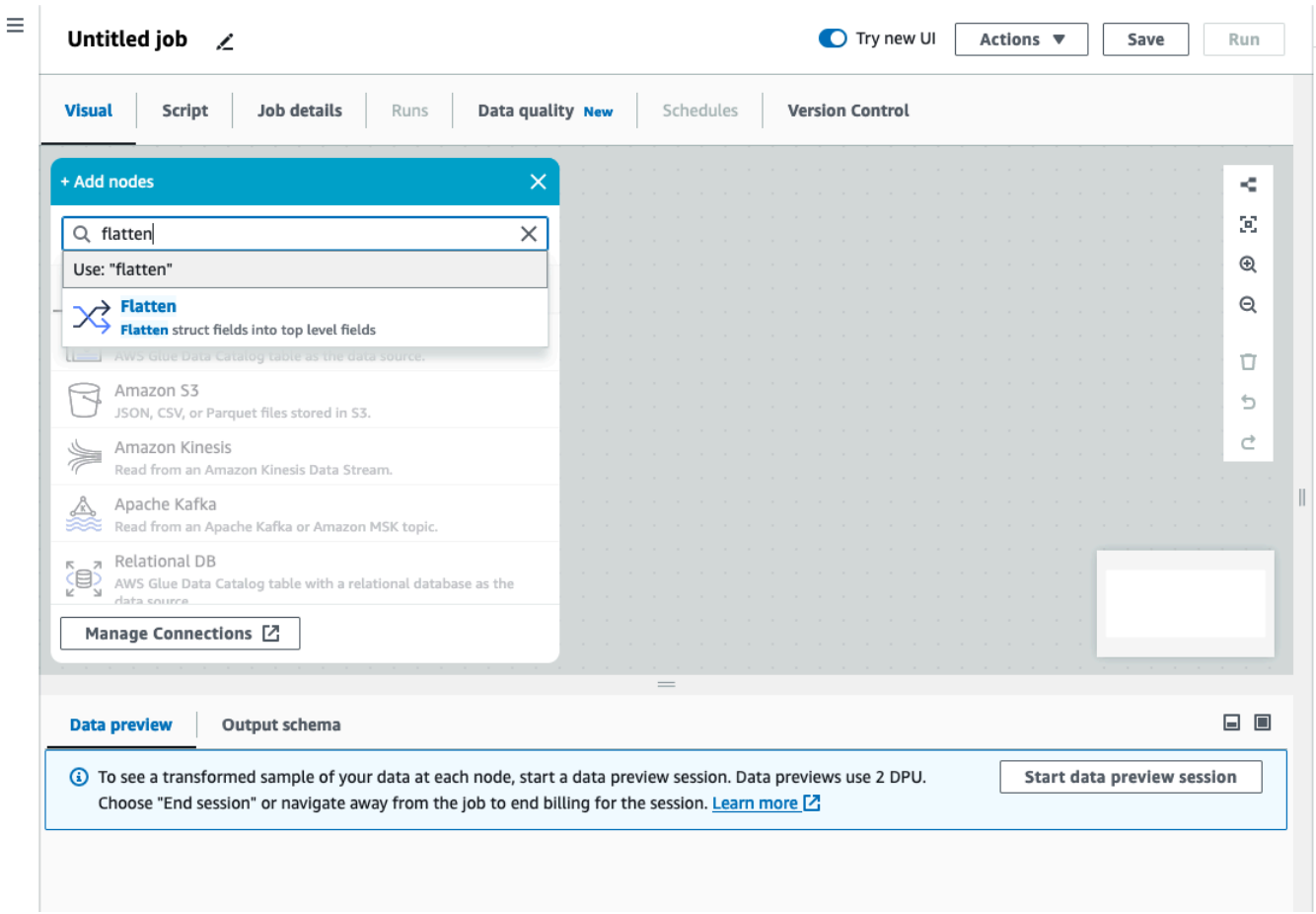
Aplana los campos de las estructuras anidadas en los datos para que se conviertan en campos de nivel superior. Los nuevos campos se denominan con el nombre del campo precedido de los nombres de los campos de la estructura que lo integran, separados por puntos.

Por ejemplo, si los datos tienen un campo de tipo de estructura llamado “phone_numbers”, que entre otros campos tiene uno del tipo “Struct” denominado “home_phone” con dos campos: “country_code” y “number”. Una vez aplanados, estos dos campos pasarán a ser campos de nivel superior denominados: “phone_numbers.home_phone.country_code” y “phone_numbers.home_phone.number”, respectivamente.

Para agregar un nodo de transformación de Aplanado al diagrama de trabajo

1. Abra el panel de recursos y elija la pestaña Transformación y, a continuación, Flatten para agregar una nueva transformación al diagrama de trabajo. También puede utilizar la barra de

búsqueda a través del ingreso de “Flatten” y, a continuación, hacer clic en el nodo Flatten. El nodo seleccionado en el momento de agregar el nodo será el nodo principal.



2. (Opcional) En la pestaña Propiedades del nodo, puede ingresar un nombre para el nodo en el diagrama de trabajo. Si todavía no se ha seleccionado un nodo principal, elija un nodo de la lista Node parents (Nodos principales) para utilizar como origen de entrada para la transformación.
3. (Opcional) En la pestaña Transformación, puede limitar el nivel máximo de anidamiento para aplanar. Por ejemplo, si establece ese valor en 1 significa que solo se aplanarán las estructuras de nivel superior. Si se establece el máximo en 2, se aplanarán el nivel superior y las estructuras que se encuentran justo debajo de este.

Agregar una columna UUID

Al agregar una columna UUID (Identificador único universal), se le asignará una cadena única de 36 caracteres a cada fila.

Para agregar un nodo de transformación de UUID en el diagrama de trabajo

1. Abra el panel de recursos y, luego, elija UUID para agregar una nueva transformación al diagrama de trabajo. El nodo seleccionado en el momento de agregar el nodo será el nodo principal.
2. (Opcional) En la pestaña Propiedades del nodo, puede ingresar un nombre para el nodo en el diagrama de trabajo. Si todavía no se ha seleccionado un nodo principal, elija un nodo de la lista Node parents (Nodos principales) para utilizar como origen de entrada para la transformación.
3. (Opcional) En la pestaña Transformación, puede personalizar el nombre de la nueva columna. De forma predeterminada, se denominará “uuid”.

Agregar una columna de identificador

Asigne un identificador numérico para cada fila del conjunto de datos.

Para agregar un nodo de transformación de Identificador en el diagrama de trabajo

1. Abra el panel de recursos y, luego, elija Identificador para agregar una nueva transformación al diagrama de trabajo. El nodo seleccionado en el momento de agregar el nodo será el nodo principal.
2. (Opcional) En la pestaña Propiedades del nodo, puede ingresar un nombre para el nodo en el diagrama de trabajo. Si todavía no se ha seleccionado un nodo principal, elija un nodo de la lista Node parents (Nodos principales) para utilizar como origen de entrada para la transformación.
3. (Opcional) En la pestaña Transformación, puede personalizar el nombre de la nueva columna. De forma predeterminada, se llamará “id”.
4. (Opcional) Si el trabajo procesa y almacena los datos de forma incremental, debe evitar que se vuelvan a utilizar los mismos id entre las ejecuciones del trabajo.

   En la pestaña Transformación, marque la opción de la casilla de verificación único. Incluirá la marca temporal del trabajo en el identificador, lo que lo hará único entre varias ejecuciones. Para permitir el número mayor, la columna, en lugar tipo long, será un decimal.

Convertir una columna al tipo de marca temporal

Puede utilizar la transformación marca temporal para cambiar el tipo de datos de una columna numérica o de cadena a una de marca temporal, de modo que pueda almacenarse con ese tipo de datos o aplicarse a otras transformaciones que requieran una marca temporal.

Para agregar un nodo de transformación de marca temporal al diagrama de trabajo

1. Abra el panel Recursos y, luego, elija Marca temporal para agregar una nueva transformación al diagrama de trabajo. El nodo seleccionado en el momento de agregar el nodo será el nodo principal.
2. (Opcional) En la pestaña Propiedades del nodo, puede ingresar un nombre para el nodo en el diagrama de trabajo. Si todavía no se ha seleccionado un nodo principal, elija un nodo de la lista Node parents (Nodos principales) para utilizar como origen de entrada para la transformación.
3. En la pestaña Transformación, introduzca el nombre de la columna que se va a convertir.
4. En la pestaña Transformación, defina cómo analizar la columna seleccionada mediante la elección del tipo.

   Si el valor es un número, se puede expresar en segundos (marca temporal de Unix/Python), milisegundos o microsegundos, elija la opción correspondiente.

   Si el valor es una cadena con formato, elija el tipo “iso”, la cadena debe ajustarse a una de las variantes del formato ISO, por ejemplo: “2022-11-02T14:40:59.915Z”.

   Si desconoce el tipo en este momento o si diferentes filas utilizan tipos diferentes, puede elegir la opción “detección automática” y el sistema hará su mejor estimación, con un pequeño costo en el rendimiento.
5. (Opcional) En la pestaña Transformación, en lugar de convertir la columna seleccionada, puede crear una nueva y conservar la original al asignar un nombre para la nueva columna.

Convertir una columna con marca temporal en una cadena con formato

Formatee una columna de marca temporal en una cadena basada en un patrón. Puede utilizar Formatear marca temporal para obtener la fecha y la hora como una cadena con el formato deseado. Puede definir el formato con la Sintaxis de fecha de Spark, así como la mayoría de los Códigos de fecha de Python.

Por ejemplo, si quiere que la cadena de fecha tenga el formato “2023-01-01 00:00”, puede definir dicho formato con la sintaxis de Spark como “yyyy-MM-dd HH:mm” o los códigos de fecha equivalentes de Python como “%Y-%m-%d %H:%M”

Para agregar un nodo de transformación de formateo de marca temporal al diagrama de trabajo

1. Abra el panel de recursos y, a continuación, elija Formatear marca temporal para agregar una nueva transformación al diagrama de trabajo. El nodo seleccionado en el momento de agregar el nodo será el nodo principal.
2. (Opcional) En la pestaña Propiedades del nodo, puede ingresar un nombre para el nodo en el diagrama de trabajo. Si todavía no se ha seleccionado un nodo principal, elija un nodo de la lista Node parents (Nodos principales) para utilizar como origen de entrada para la transformación.
3. En la pestaña Transformación, introduzca el nombre de la columna que se va a convertir.
4. En la pestaña Transformación, introduzca el patrón de Formato de marca temporal que desea utilizar, expresado mediante la sintaxis de fecha de Spark o los códigos de fecha de Python.
5. (Opcional) En la pestaña Transformación, en lugar de convertir la columna seleccionada, puede crear una nueva y conservar la original al asignar un nombre para la nueva columna.

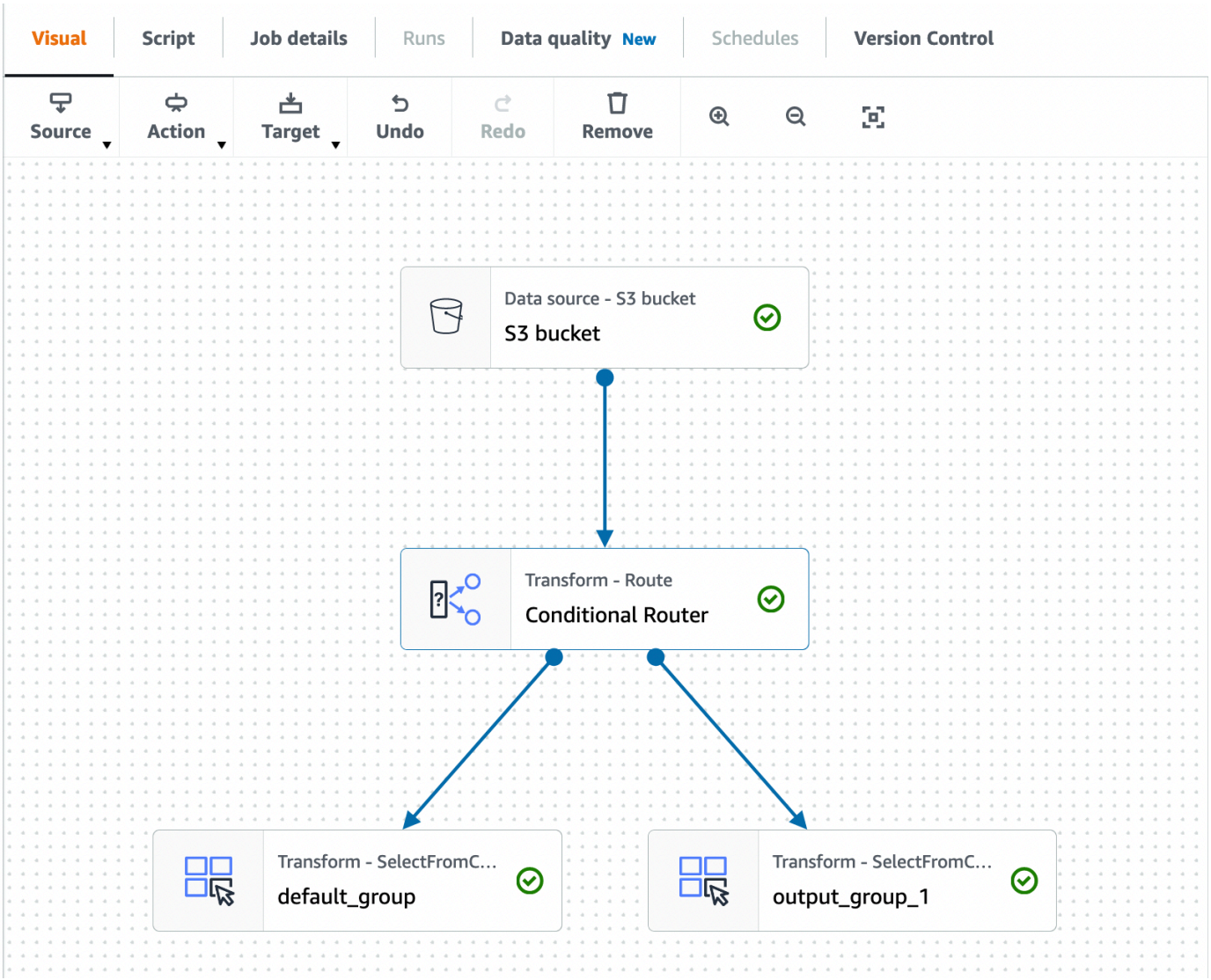
Creación de una transformación de Enrutador condicional

La transformación del Enrutador condicional permite aplicar múltiples condiciones a los datos entrantes. Cada fila de los datos entrantes se evalúa mediante una condición de filtro de grupo y se procesa en su grupo correspondiente. Si una fila cumple con más de una condición de filtro de grupo, la transformación pasa la fila a varios grupos. Si una fila no cumple con ninguna condición, se puede eliminar o enrutar a un grupo de salida predeterminado.

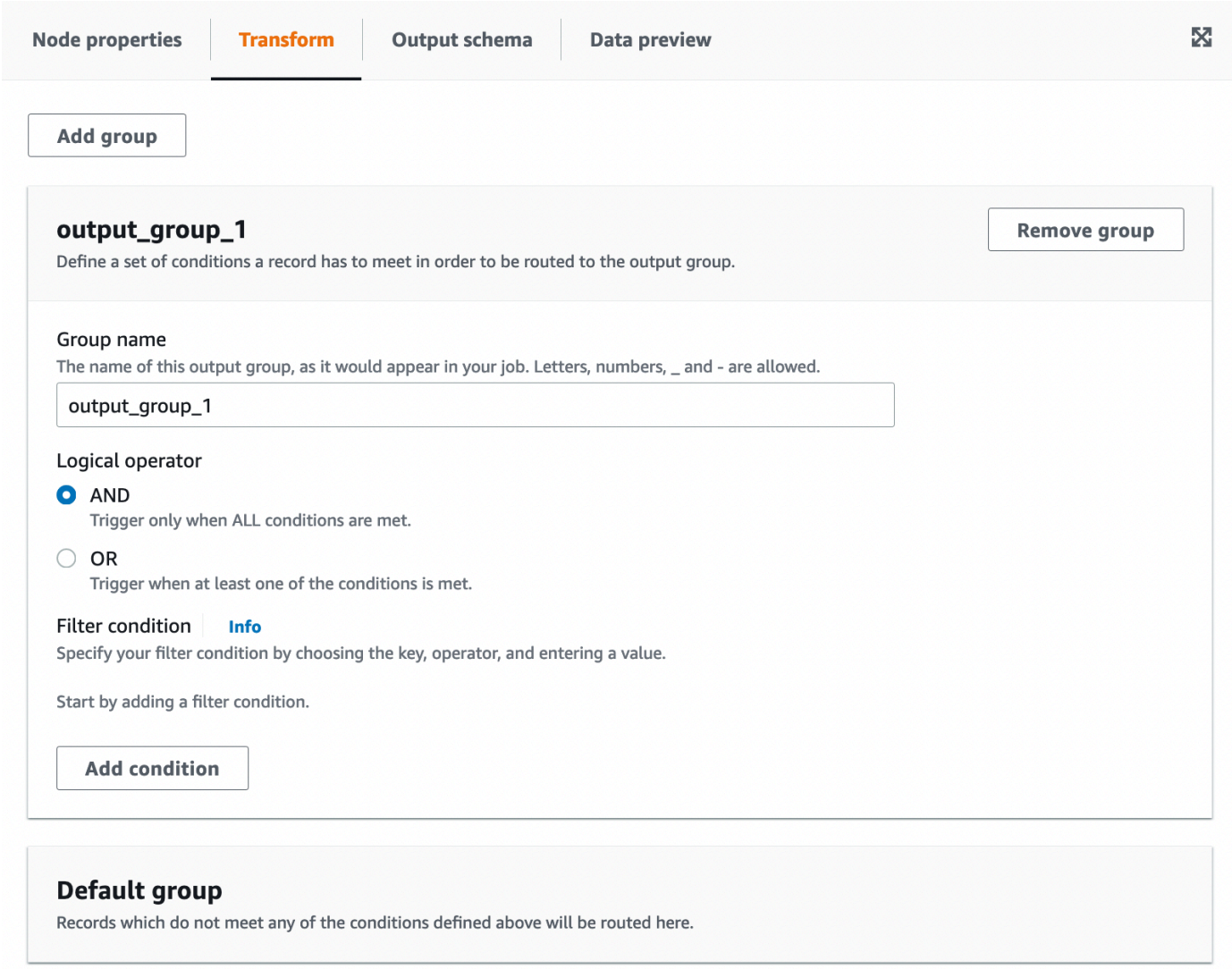
Esta transformación es similar a la transformación de filtro, pero es útil para los usuarios que desean probar los mismos datos de entrada en múltiples condiciones.

Para agregar una transformación de Enrutador condicional:

1. Elija un nodo en el que realizará la transformación de enrutador condicional. Puede ser un nodo de origen o una transformación distinta.
2. Elija Acción y, a continuación, utilice la barra de búsqueda para buscar y elegir “Enrutador condicional”. Se agrega una transformación de Enrutador condicional junto con dos nodos de salida. Un nodo de salida, “Grupo predeterminado”, contiene registros que no cumplen ninguna de las condiciones definidas en los otros nodos de salida. El grupo predeterminado no se puede editar.

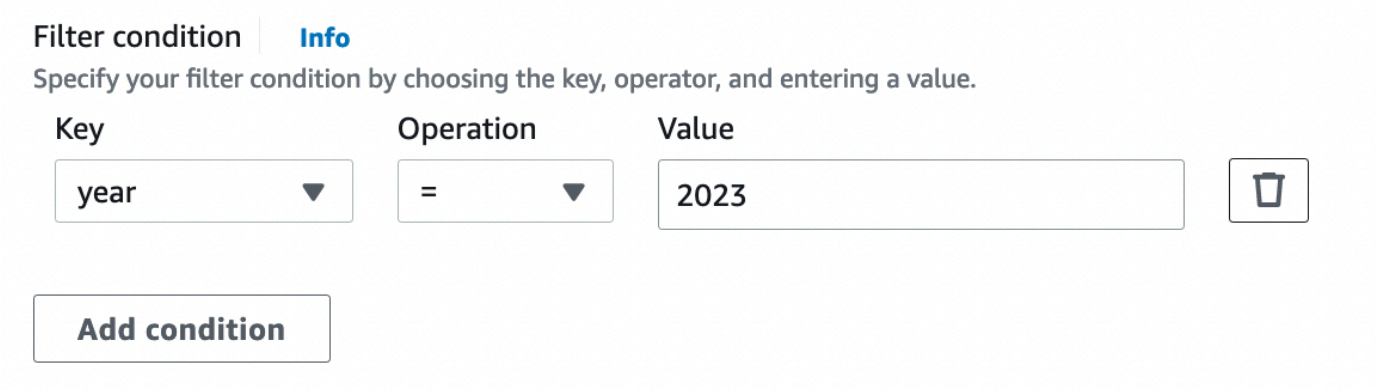


Puede agregar grupos de salida adicionales si selecciona Agregar grupo. Puede asignar un nombre a cada grupo de resultado y agregar condiciones de filtro y un operador lógico.



3. Cambie el nombre del grupo de salida al introducir un nombre nuevo para el grupo. AWS Glue Studio asignará de forma automática un nombre a los grupos (por ejemplo, “output_group_1”).
4. Elija un operador lógico (AND, OR) y agregue una Condición de filtro mediante la especificación de la Clave, la Operación y el Valor. Los operadores lógicos permiten implementar más de una condición de filtro y aplicar el operador lógico en cada condición de filtro que especifique.

   Cuando especifique la clave, puede elegir entre las claves disponibles en el esquema. Luego, puede elegir la operación disponible según el tipo de clave que seleccionó. Por ejemplo, si el tipo de clave es “cadena”, la operación disponible para elegir es “coincidencias”.



5. Ingrese el valor en el campo Valor. Elija Agregar condición para agregar condiciones de filtro adicionales. Para eliminar las condiciones de filtro, elija el icono de la papelera.

Uso de la transformación Concatenar columnas para agregar columnas

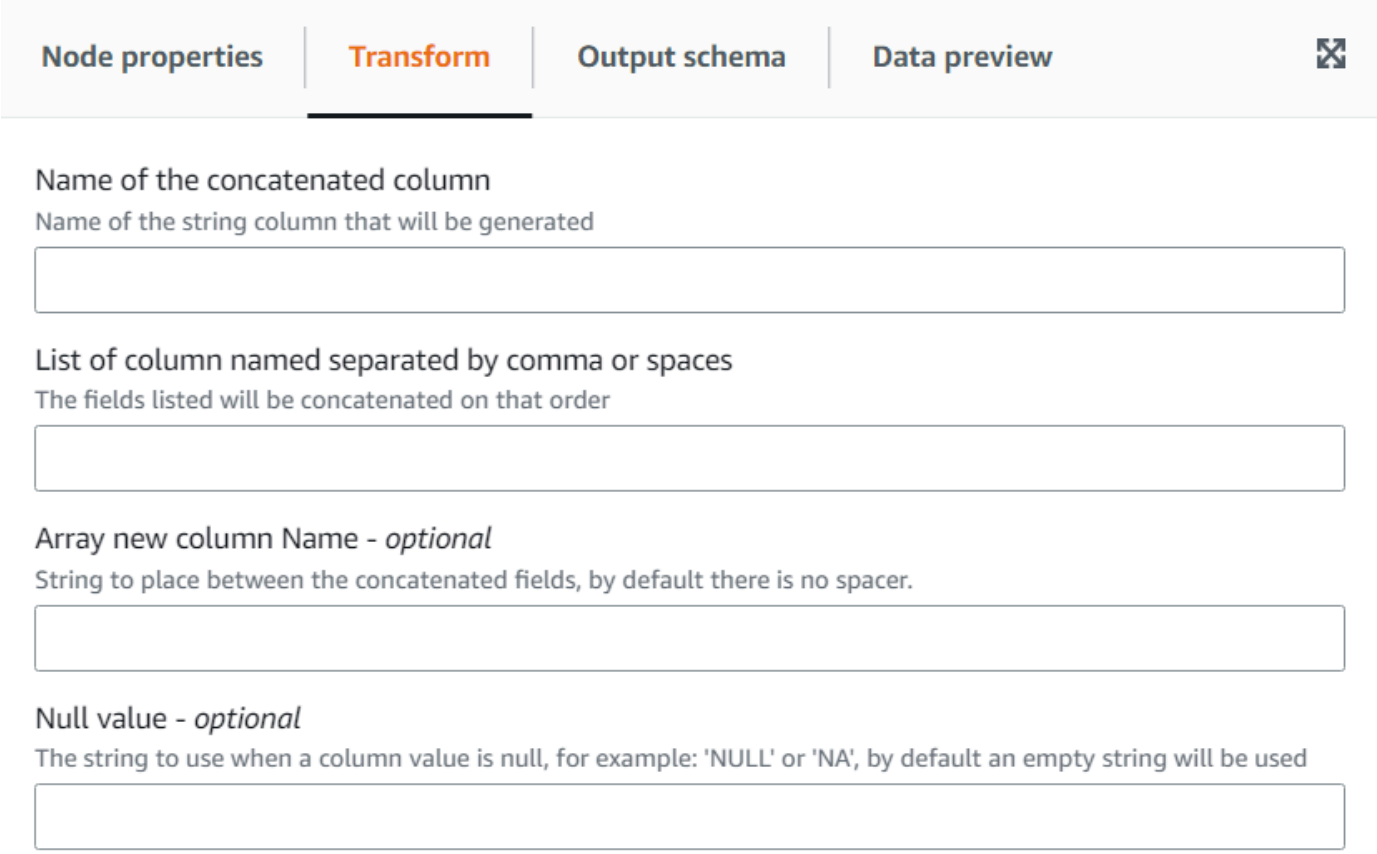
La transformación de concatenar permite crear una nueva columna de cadena mediante la utilización de los valores de otras columnas con un espaciador opcional. Por ejemplo, si definimos una columna concatenada “date” como la concatenación de “año”, “mes” y “día” (en ese orden) con “-” como espaciador, obtendremos:

day	month	year	date
01	01	2020	2020-01-01
02	01	2020	2020-01-02
03	01	2020	2020-01-03
04	01	2020	2020-01-04

Para agregar una transformación de Concatenar:

1. Abre el panel de recursos. A continuación, elija Concatenar columnas para agregar una nueva transformación al diagrama de trabajo. El nodo seleccionado en el momento de agregar el nodo será el nodo principal.
2. (Opcional) En la pestaña Propiedades del nodo, puede ingresar un nombre para el nodo en el diagrama de trabajo. Si todavía no se ha seleccionado un nodo principal, elija un nodo de la lista Nodos principales para utilizar como origen de entrada para la transformación.

3. En la pestaña Transformación, ingrese el nombre de la columna que contendrá la cadena concatenada, así como las columnas que se concatenarán. El orden en el que selecciones las columnas en el menú desplegable será el orden en que se usen.



4. Espaciador (opcional): introduce una cadena para colocarla entre los campos concatenados. No hay un espaciador predeterminado.
5. Valor nulo (opcional): ingrese una cadena para utilizarla cuando el valor de una columna sea nulo. De forma predeterminada, en los casos en que las columnas tienen el valor “NULO” o “NA”, se utiliza una cadena vacía.

Uso de la transformación de cadena dividida para dividir una columna de cadena

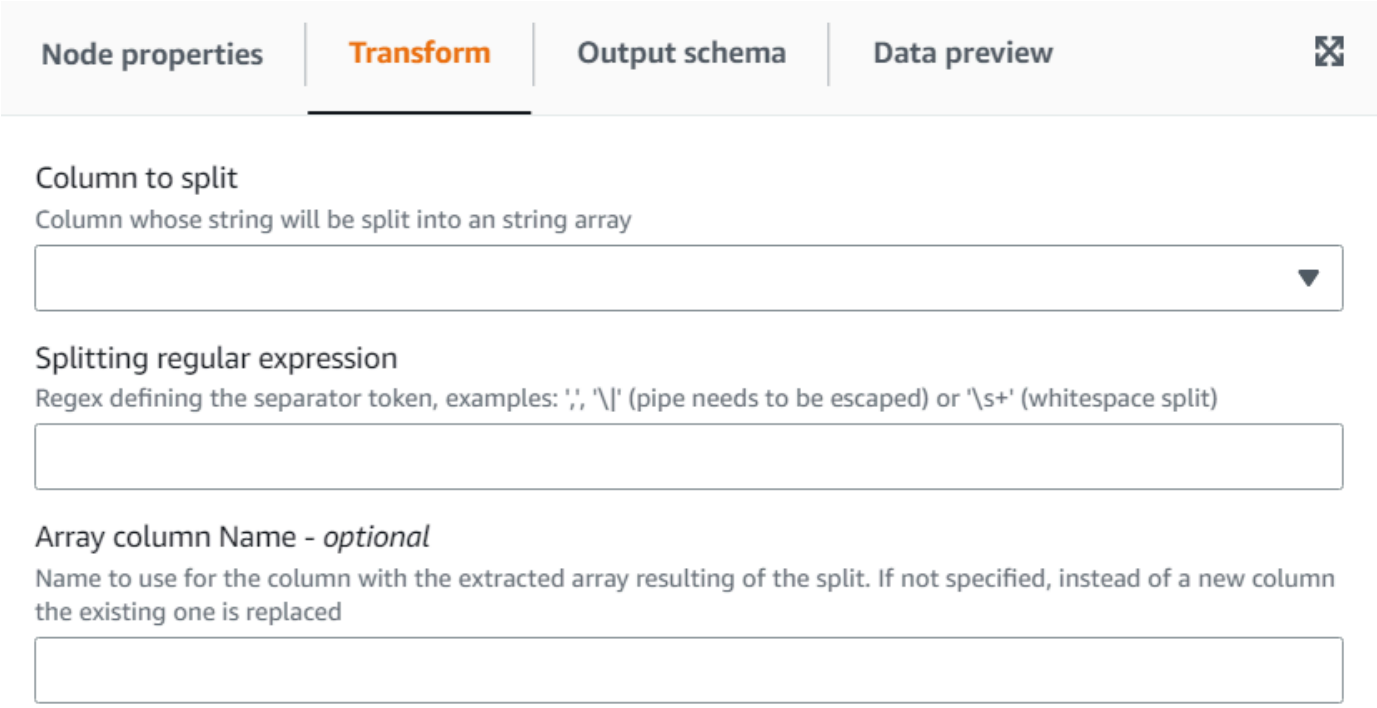
La transformación de cadena dividida permite dividir una cadena en una matriz de símbolos mediante una expresión regular para definir cómo se realiza la división. Luego, puede mantener la columna como un tipo de matriz o aplicar una transformación de matriz a columnas después de esta, para extraer los valores de la matriz en los campos de nivel superior, suponiendo que cada token tenga un significado que conozcamos de antemano. Además, si el orden de los símbolos es irrelevante (por ejemplo, un conjunto de categorías), puedes usar la transformación Expansión para generar una fila independiente para cada valor.

Por ejemplo, se puede dividir la columna “categorías” mediante una coma como patrón para agregar una columna “categories_arr”.

product_id	categories	categories_arr
1	deportes, invierno	[deportes, invierno]
2	jardín, herramientas	[jardín, herramientas]
3	video juegos	[video juegos]
4	juego, juego de mesa, social	[juego, juego de mesa, social]

Para agregar una transformación de cadena dividida, realice lo siguiente:

1. Abra el panel de recursos y, luego, elija Cadena dividida para agregar una nueva transformación al diagrama de trabajo. El nodo seleccionado en el momento de agregar el nodo será el nodo principal.
2. (Opcional) En la pestaña Propiedades del nodo, puede ingresar un nombre para el nodo en el diagrama de trabajo. Si todavía no se ha seleccionado un nodo principal, elija un nodo de la lista Nodos principales para utilizar como origen de entrada para la transformación.
3. En la pestaña Transformación, elija la columna que desee dividir e ingrese el patrón que se utilizará para dividir la cadena. En la mayoría de los casos, solo tiene que ingresar los caracteres, a menos que tengan un significado especial como expresión regular y deban ocultarse. Los caracteres que hay que ocultar son: `\.[]{}()<>*+-=!?^$|` mediante la adición de una barra invertida delante del carácter. Por ejemplo, si desea separarlos por un punto ('.'), debe ingresar `\.`. Sin embargo, una coma no tiene un significado especial y se puede especificar tal cual: `,`.



4. (Opcional) Si desea conservar la columna de cadena original, puede ingresar un nombre para una nueva columna de matriz, de manera que se mantenga tanto la columna de cadena original como la nueva columna de matriz tokenizada.

Uso de la transformación de Matriz a columnas para extraer los elementos de una matriz en columnas de nivel superior

La transformación de Matriz a columnas permite extraer algunos o todos los elementos de una columna de tipo matriz en nuevas columnas. La transformación llenará las nuevas columnas tanto como sea posible si la matriz tiene valores suficientes para extraerlos y, opcionalmente, tomará los elementos en las posiciones especificadas.

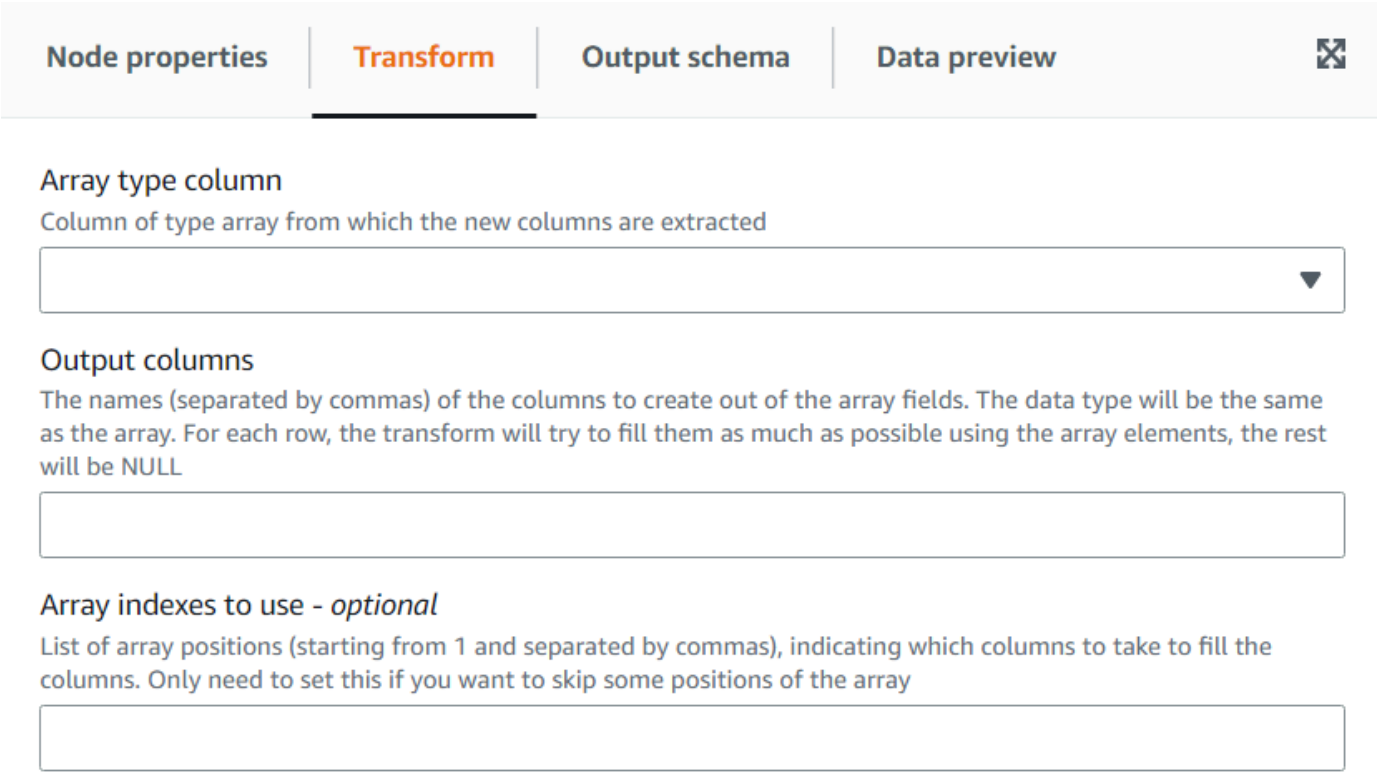
Por ejemplo, si tiene una columna de matriz llamada “subred”, que fue el resultado de aplicar la transformación “cadena dividida” en una subred ip v4, puede extraer la primera y la cuarta posición en las nuevas columnas “first_octect” y “forth_octect”. El resultado de la transformación en este ejemplo sería (observe que las dos últimas filas tienen matrices más cortas de lo esperado):

subred	first_octect	fourth_octect
[54, 240, 197, 238]	54	238

subred	first_octect	fourth_octect
[192, 168, 0, 1]	192	1
[192, 168]	192	
[]		

Para agregar una transformación de Matriz a columnas:

1. Abra el panel de recursos y elija Matriz a columnas para agregar una nueva transformación al diagrama de trabajo. El nodo seleccionado en el momento de agregar el nodo será el nodo principal.
2. (Opcional) En la pestaña Propiedades del nodo, puede ingresar un nombre para el nodo en el diagrama de trabajo. Si todavía no se ha seleccionado un nodo principal, elija un nodo de la lista Nodos principales para utilizar como origen de entrada para la transformación.
3. En la pestaña Transformación, elija la columna de matriz a extraer e ingrese la lista de columnas nuevas para los tokens extraídos.



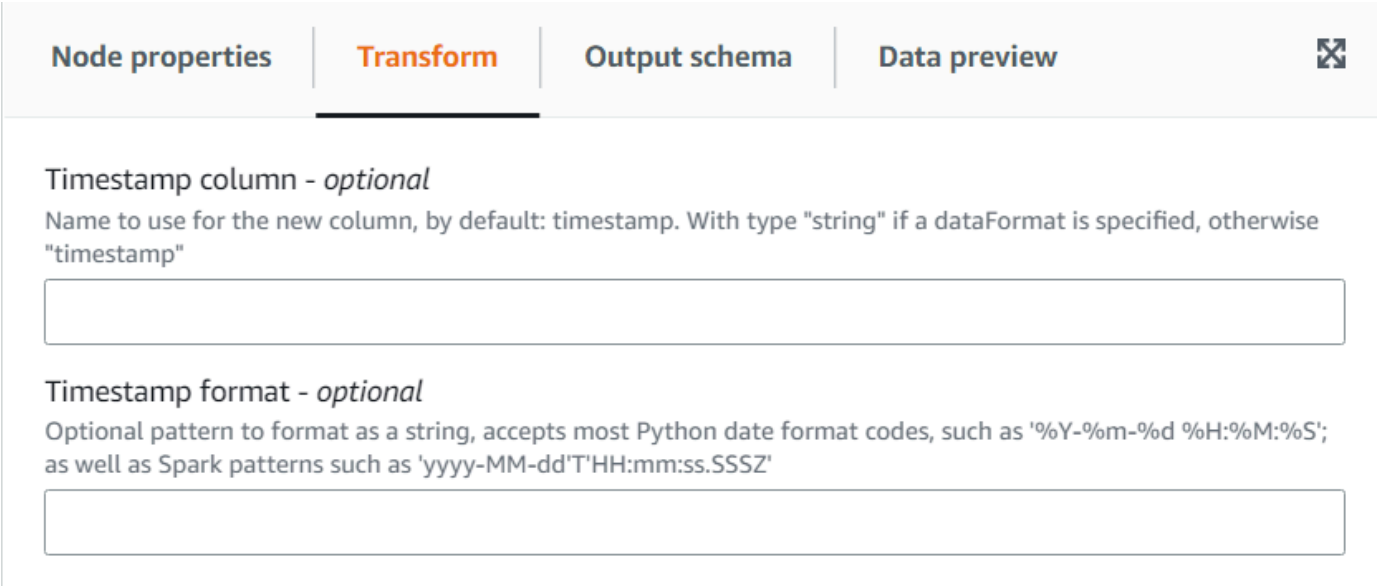
4. (Opcional) Si no desea utilizar los tokens de la matriz para asignarlos a las columnas, puede especificar los índices que se utilizarán, que se asignarán a la lista de columnas en el mismo orden especificado. Por ejemplo, si las columnas de salida son “columna1, columna2, columna3” y los índices “4, 1, 3”, el cuarto elemento de la matriz irá a la columna1, el primero a la columna2 y el tercero a la columna3 (si la matriz es más corta que el número de índice, se establecerá un valor NULO).

Uso de la transformación Agregar marca de tiempo actual

La transformación Agregar marca de tiempo actual permite marcar las filas con la hora en la que se procesaron los datos. Esto resulta útil para fines de auditoría o para realizar un seguimiento de la latencia en la canalización de datos. Puede agregar esta nueva columna como un tipo de datos de marca de tiempo o una cadena formateada.

Para agregar una transformación de tipo Add Current Timestamp:

1. Abra el panel de recursos y, a continuación, elija Agregar marca de tiempo actual para agregar una nueva transformación al diagrama de trabajo. El nodo seleccionado en el momento de agregar el nodo será el nodo principal.
2. (Opcional) En la pestaña Propiedades del nodo, puede ingresar un nombre para el nodo en el diagrama de trabajo. Si todavía no se ha seleccionado un nodo principal, elija un nodo de la lista Nodos principales para utilizar como origen de entrada para la transformación.



3. (Opcional) En la pestaña Transformación, ingrese un nombre personalizado para la nueva columna y un formato si prefiere que la columna sea una cadena de fecha con formato.

Uso de la transformación de rotar filas a columnas

La transformación filas dinámicas a columnas permite agregar una columna numérica mediante la rotación de valores únicos en las columnas seleccionadas para convertirlas en nuevas columnas (si se seleccionan varias columnas, los valores se concatenan para dar nombre a las nuevas columnas). De esta forma, las filas se consolidan y hay más columnas con agregaciones parciales para cada valor único. Por ejemplo, si tiene este conjunto de datos de ventas por mes y país (ordenado para que sea más fácil de ilustrar):

año	mes	país	importe
2020	Ene	uk	32
2020	Ene	de	42
2020	Ene	us	64
2020	Feb	uk	67
2020	Feb	de	4
2020	Feb	de	7
2020	Feb	us	6
2020	Feb	us	12
2020	Ene	us	90

Si cambia el importe y el país como columnas de agregación, se crean nuevas columnas a partir de la columna del país original. En la siguiente tabla, tiene nuevas columnas para de, uk y us en lugar de la columna de país.

año	mes	de	uk	us
2020	Ene	42	32	64
2020	Ene	11	67	18

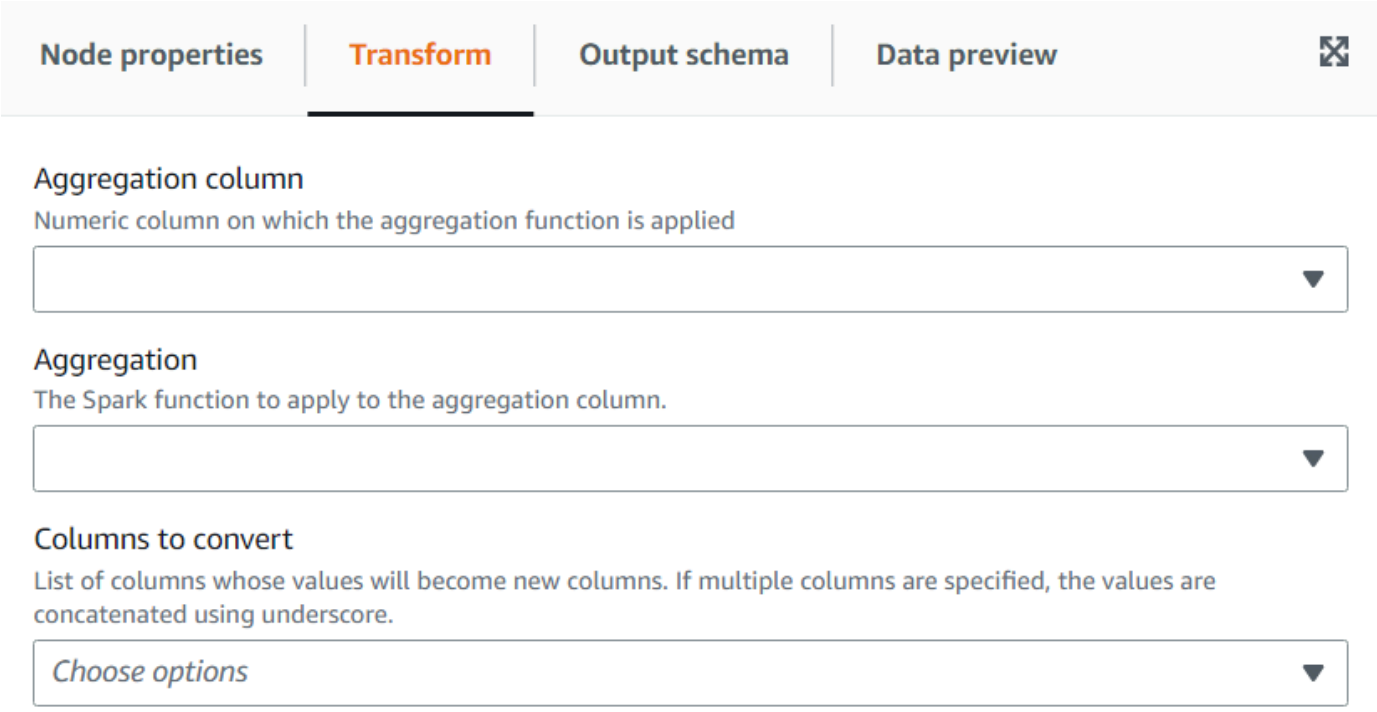
año	mes	de	uk	us
2021	Ene			90

Si, por el contrario, desea cambiar el mes y el condado, obtendrá una columna para cada combinación de los valores de esas columnas:

year	Jan_DE	Jan_uk	Jan_us	Feb_de	Feb_UK	Feb_US
2020	42	32	64	11	67	18
2021			90			

Para agregar una transformación de filas dinámicas a columnas, realice lo siguiente:

1. Abra el panel de recursos y, luego, elija Filas dinámicas a columnas para agregar una nueva transformación al diagrama de trabajo. El nodo seleccionado en el momento de agregar el nodo será el nodo principal.
2. (Opcional) En la pestaña Propiedades del nodo, puede ingresar un nombre para el nodo en el diagrama de trabajo. Si todavía no se ha seleccionado un nodo principal, elija un nodo de la lista Nodos principales para utilizar como origen de entrada para la transformación.
3. En la pestaña Transformación, elija la columna numérica que se agregará para generar los valores de las nuevas columnas, la función de agregación que desee aplicar y las columnas para convertir sus valores únicos en columnas nuevas.



Uso de la transformación de anulación de rotación de columnas a filas

La transformación Anular rotación permite convertir columnas en valores de nuevas columnas, lo que genera una fila para cada valor único. Si bien es lo opuesto a rotar, se debe tener en cuenta que no es equivalente, ya que no puede separar filas con valores idénticos que se hayan agregado ni dividir combinaciones en las columnas originales (se puede hacerlo más adelante mediante una transformación dividida). Por ejemplo, si tiene la siguiente carga:

año	mes	de	uk	us
2020	Ene	42	32	64
2020	Feb	11	67	18
2021	Ene			90

Se puede anular la rotación de las columnas “de”, “uk” y “us” y convertirlas en una columna “país” con el valor “importe” y obtener lo siguiente (ordenado aquí a modo ilustrativo):

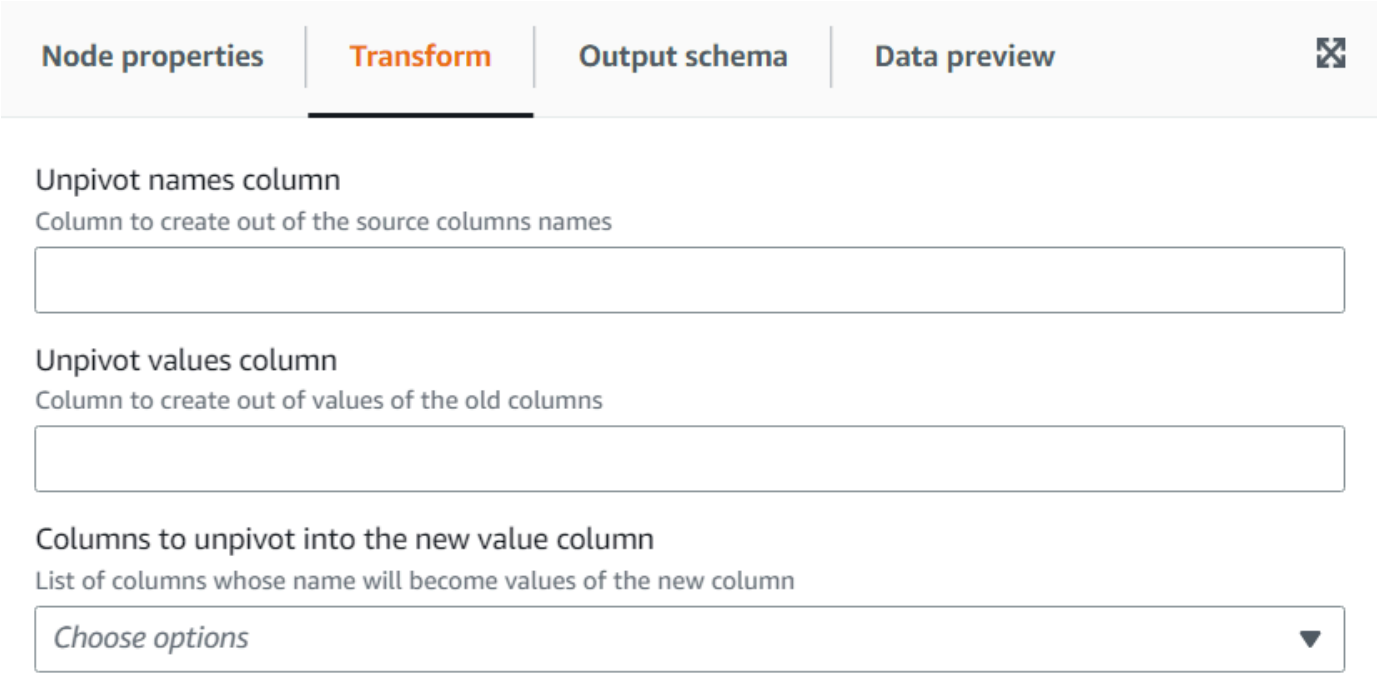
año	mes	país	importe
2020	Ene	uk	32
2020	Ene	de	42
2020	Ene	us	64
2020	Feb	uk	67
2020	Feb	de	11
2020	Feb	us	18
2021	Ene	us	90

Observe que las columnas que tienen un valor NULO (“de” y “uk” de enero de 2021) no se generan de forma predeterminada. Puede activar esa opción para obtener:

año	mes	país	importe
2020	Ene	uk	32
2020	Ene	de	42
2020	Ene	us	64
2020	Feb	uk	67
2020	Feb	de	11
2020	Feb	us	18
2021	Ene	us	90
2021	Ene	de	
2021	Ene	uk	

Para agregar una transformación de anulación de rotación de columnas a filas, realice lo siguiente:

1. Abra el panel de recursos y, luego, elija Anular rotación de columnas en filas para agregar una nueva transformación al diagrama de trabajo. El nodo seleccionado en el momento de agregar el nodo será el nodo principal.
2. (Opcional) En la pestaña Propiedades del nodo, puede ingresar un nombre para el nodo en el diagrama de trabajo. Si todavía no se ha seleccionado un nodo principal, elija un nodo de la lista Nodos principales para utilizar como origen de entrada para la transformación.
3. En la pestaña Transformación, ingrese las nuevas columnas que se van a crear para que contengan los nombres y valores de las columnas seleccionadas para anular la rotación.

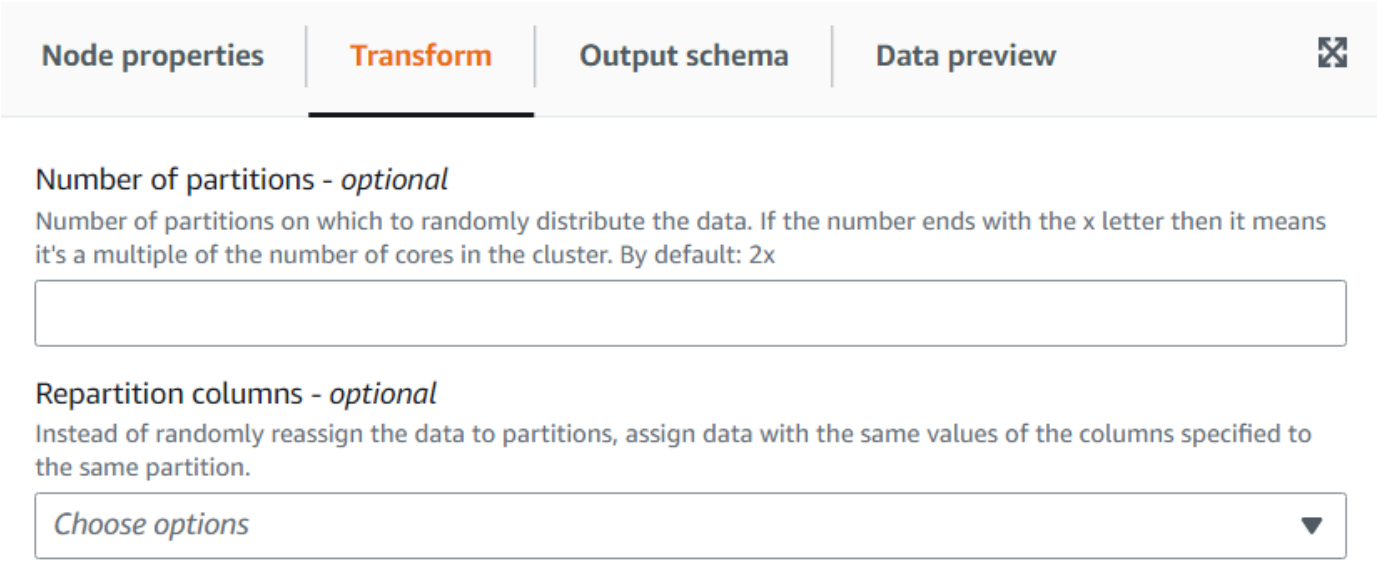


Uso de la transformación de Procesamiento de autobalance para optimizar el tiempo de ejecución

La transformación de Procesamiento de autobalance redistribuye los datos entre los trabajadores para mejorar el rendimiento. Esto ayuda en los casos en que los datos están desequilibrados o, tal como provienen del origen, no permiten un procesamiento paralelo suficiente. Esto es común cuando el origen está comprimido con gzip o es JDBC. La redistribución de los datos tiene un costo de rendimiento modesto, por lo que es posible que la optimización no siempre compense ese esfuerzo si los datos ya estaban bien equilibrados. En la imagen inferior, la transformación utiliza la repartición de Apache Spark para reasignar datos de forma aleatoria entre un número de particiones óptimo

para la capacidad del clúster. Para los usuarios avanzados, es posible ingresar varias particiones de forma manual. Además, se puede utilizar para optimizar la escritura de tablas particionadas al reorganizar los datos en función de columnas específicas. Esto da como resultado archivos de salida más consolidados.

1. Abra el panel de recursos y, a continuación, elija Procesamiento de autobalance para agregar una nueva transformación al diagrama de trabajo. El nodo seleccionado en el momento de agregar el nodo será el nodo principal.
2. (Opcional) En la pestaña Propiedades del nodo, puede ingresar un nombre para el nodo en el diagrama de trabajo. Si todavía no se ha seleccionado un nodo principal, elija un nodo de la lista Nodos principales para utilizar como origen de entrada para la transformación.
3. (Opcional) En la pestaña Transformación, puede ingresar varias particiones. En general, se recomienda dejar que el sistema decida este valor; sin embargo, puede ajustar el multiplicador o ingresar un valor específico si necesita controlarlo. Si va a guardar los datos particionados por columnas, puede elegir las mismas columnas como columnas de repartición. De esta forma, minimizará la cantidad de archivos en cada partición y evitará tener muchos archivos por partición, lo que dificultaría el rendimiento de las herramientas que consultan esos datos.



Uso de la transformación de columna derivada para combinar otras columnas

La transformación de Columna derivada permite definir una nueva columna basada en una fórmula matemática o expresión SQL en la que puede utilizar otras columnas de los datos, así como

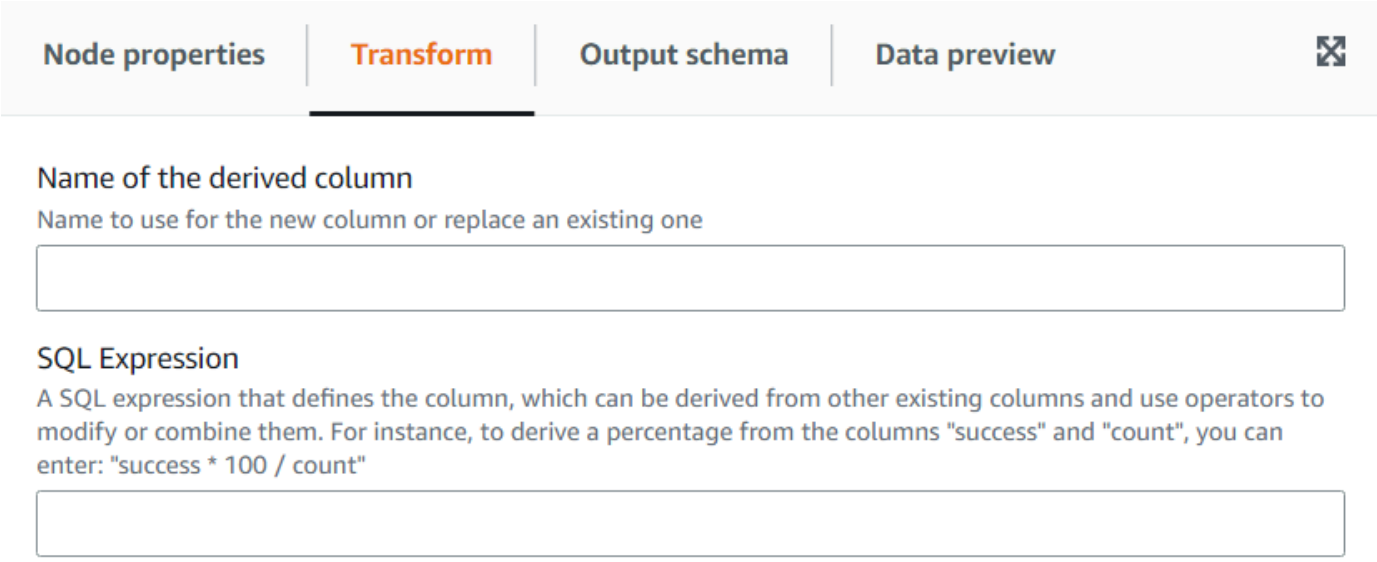
constantes y literales. Por ejemplo, para obtener una columna de “porcentaje” a partir de las columnas “éxito” y “recuento”, puede ingresar la expresión SQL: “éxito * 100 / recuento || ‘%’”.

Ejemplo de resultado:

success	count	percentage
14	100	14 %
6	20	3 %
3	40	7.5 %

Para agregar una transformación de columna derivada:

1. Abra el panel de recursos y, a continuación, elija Columna derivada para agregar una nueva transformación al diagrama de trabajo. El nodo seleccionado en el momento de agregar el nodo será el nodo principal.
2. (Opcional) En la pestaña Propiedades del nodo, puede ingresar un nombre para el nodo en el diagrama de trabajo. Si todavía no se ha seleccionado un nodo principal, elija un nodo de la lista Nodos principales para utilizar como origen de entrada para la transformación.
3. En la pestaña Transformación, ingrese el nombre de la columna y la expresión de su contenido.



Uso de la transformación de búsqueda para agregar datos coincidentes de una tabla de catálogo

La transformación de búsqueda permite agregar columnas de una tabla de catálogo definida cuando las claves coinciden con las columnas de búsqueda definidas en los datos. Esto equivale a hacer una unión exterior izquierda entre los datos y la tabla de consulta al usar como condición columnas coincidentes.

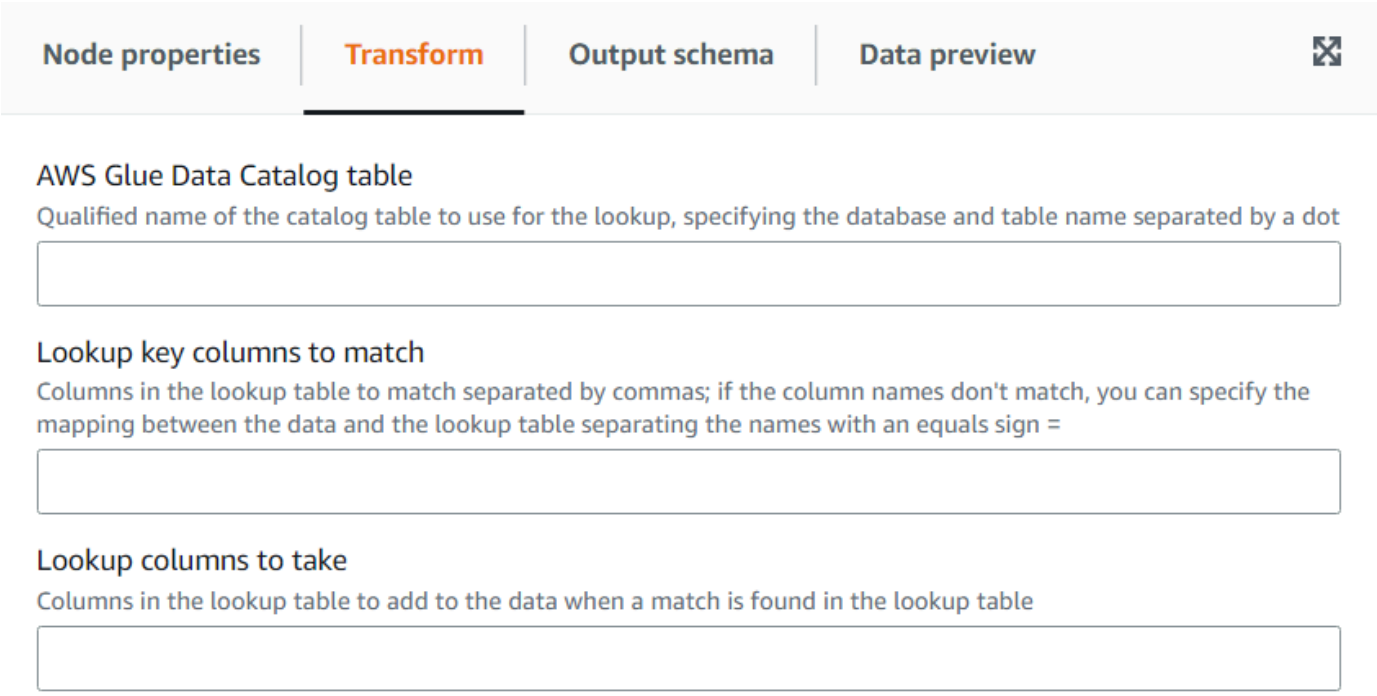
Para agregar una transformación de búsqueda, realice lo siguiente:

1. Abra el panel de recursos y, luego, elija Búsqueda para agregar una nueva transformación al diagrama de trabajo. El nodo seleccionado en el momento de agregar el nodo será el nodo principal.
2. (Opcional) En la pestaña Propiedades del nodo, puede ingresar un nombre para el nodo en el diagrama de trabajo. Si todavía no se ha seleccionado un nodo principal, elija un nodo de la lista Nodos principales para utilizar como origen de entrada para la transformación.
3. En la pestaña Transformación, ingrese el nombre completo de la tabla de catálogo que desee utilizar para realizar las búsquedas. Por ejemplo, si la base de datos es “mydb” y la tabla es “mytable”, ingrese “mydb.mytable”. A continuación, ingrese los criterios para buscar una coincidencia en la tabla de consulta, si la clave de búsqueda está compuesta. Ingrese la lista de columnas clave separadas por comas. Si una o más de las columnas clave no tienen el mismo nombre, se debe definir el mapeo de coincidencias.

   Por ejemplo, si las columnas de datos son “user_id” y “region” y en la tabla de usuarios las columnas correspondientes se denominan “id” y “region”, en el campo Columnas que deben coincidir, ingrese: “user_id=id, region”. Puede usar region=region, pero no es necesario, ya que son lo mismo.
4. Por último, ingrese las columnas que desee extraer de la fila correspondiente en la tabla de consulta para incorporarlas a los datos. Si no se encontró ninguna coincidencia, dichas columnas se establecerán en CERO.

> **Note**
>
> Debajo de la transformación de búsqueda, se utiliza una unión a la izquierda para ser eficiente. Si la tabla de búsqueda tiene una clave compuesta, asegúrese de que las columnas que deben coincidir estén configuradas para que coincidan con todas las columnas clave, de modo que solo pueda producirse una coincidencia. De lo contrario,

coincidirán varias filas de búsqueda y, como resultado, se agregarán filas adicionales para cada una de esas coincidencias.



Uso de la transformación Desglosar matriz o Mapa en filas

La transformación Desglosar permite extraer valores de una estructura anidada en filas individuales que son más fáciles de manipular. En el caso de una matriz, la transformación generará una fila para cada valor de la matriz, mediante la replicación de los valores de las demás columnas de la fila. En el caso de un mapa, la transformación generará una fila para cada entrada con la clave y el valor como columnas más cualquier otra columna de la fila.

Por ejemplo, si tenemos este conjunto de datos que tiene una columna de matriz llamada “categorías” con varios valores.

product_id	categoría
1	[deportes, invierno]
2	[jardín, herramientas]

product_id	categoría
3	[video juegos]
4	[juego, juego de mesa, social]
5	[]

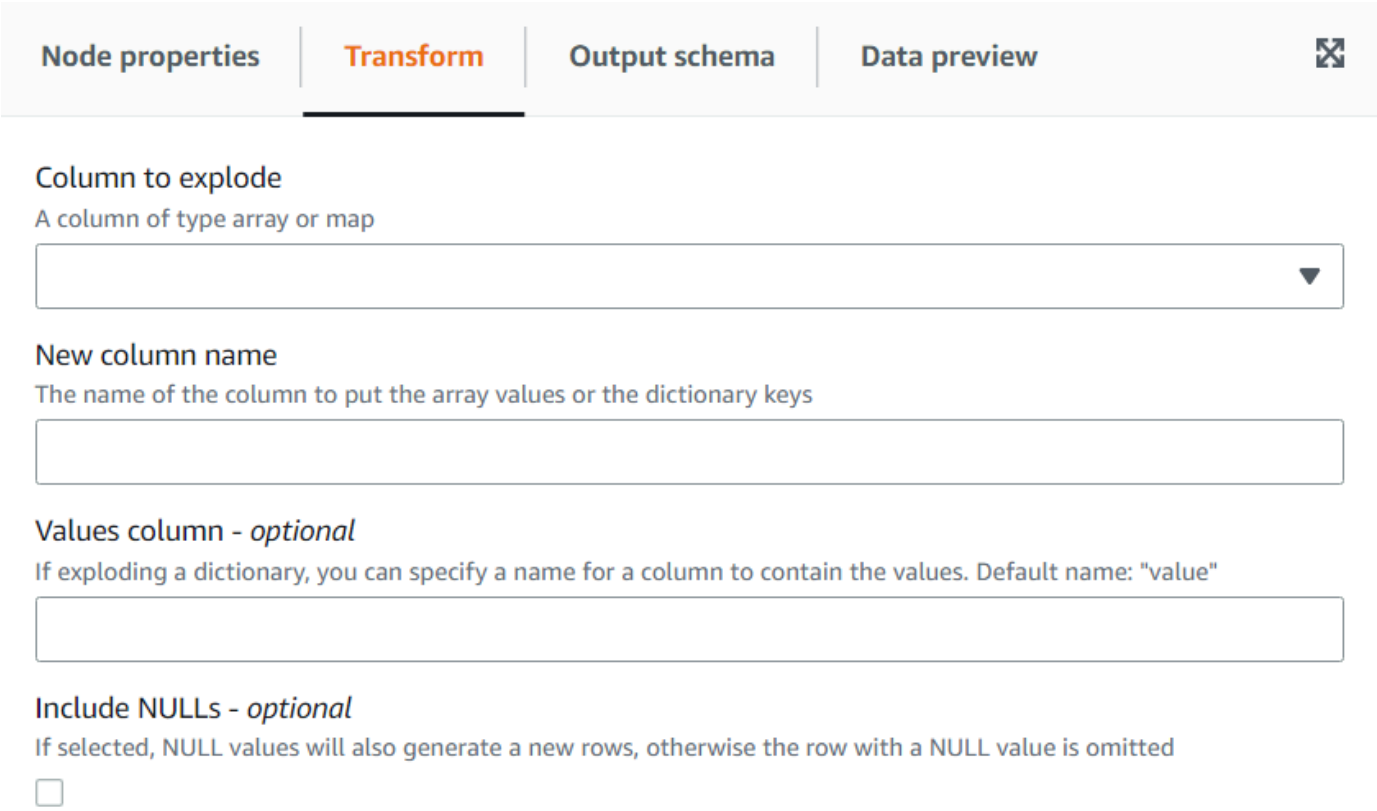
Si desglosas la columna de “categorías” en una columna con el mismo nombre, la anularás. Puede seleccionar si desea incluir los valores NULO para obtener lo siguiente (ordenados con fines ilustrativos):

product_id	categoría
1	deportes/
1	invierno
2	jardín
2	herramienta
3	video juegos
4	partido
4	juego de mesa
4	social
5	

Para agregar una transformación de Desglosar matriz o Mapa en filas:

1. Abra el panel de recursos y, a continuación, elija Desglosar matriz o Mapa en filas para agregar una nueva transformación al diagrama de trabajo. El nodo seleccionado en el momento de agregar el nodo será el nodo principal.

2. (Opcional) En la pestaña Propiedades del nodo, puede ingresar un nombre para el nodo en el diagrama de trabajo. Si todavía no se ha seleccionado un nodo principal, elija un nodo de la lista Nodos principales para utilizar como origen de entrada para la transformación.
3. En la pestaña Transformación, elija la columna que desee desglosar (debe ser de tipo matriz o mapa). A continuación, ingrese un nombre para la columna para los elementos de la matriz o los nombres de las columnas para las claves y los valores si va a descomponer un mapa.
4. (Opcional) En la pestaña Transformación, de forma predeterminada, si la columna que se va a desglosar es NULA o tiene una estructura vacía, se omitirá en el conjunto de datos desglosado. Si desea conservar la fila (con las nuevas columnas como NULAS), active la casilla “Incluir valores NULOS”.



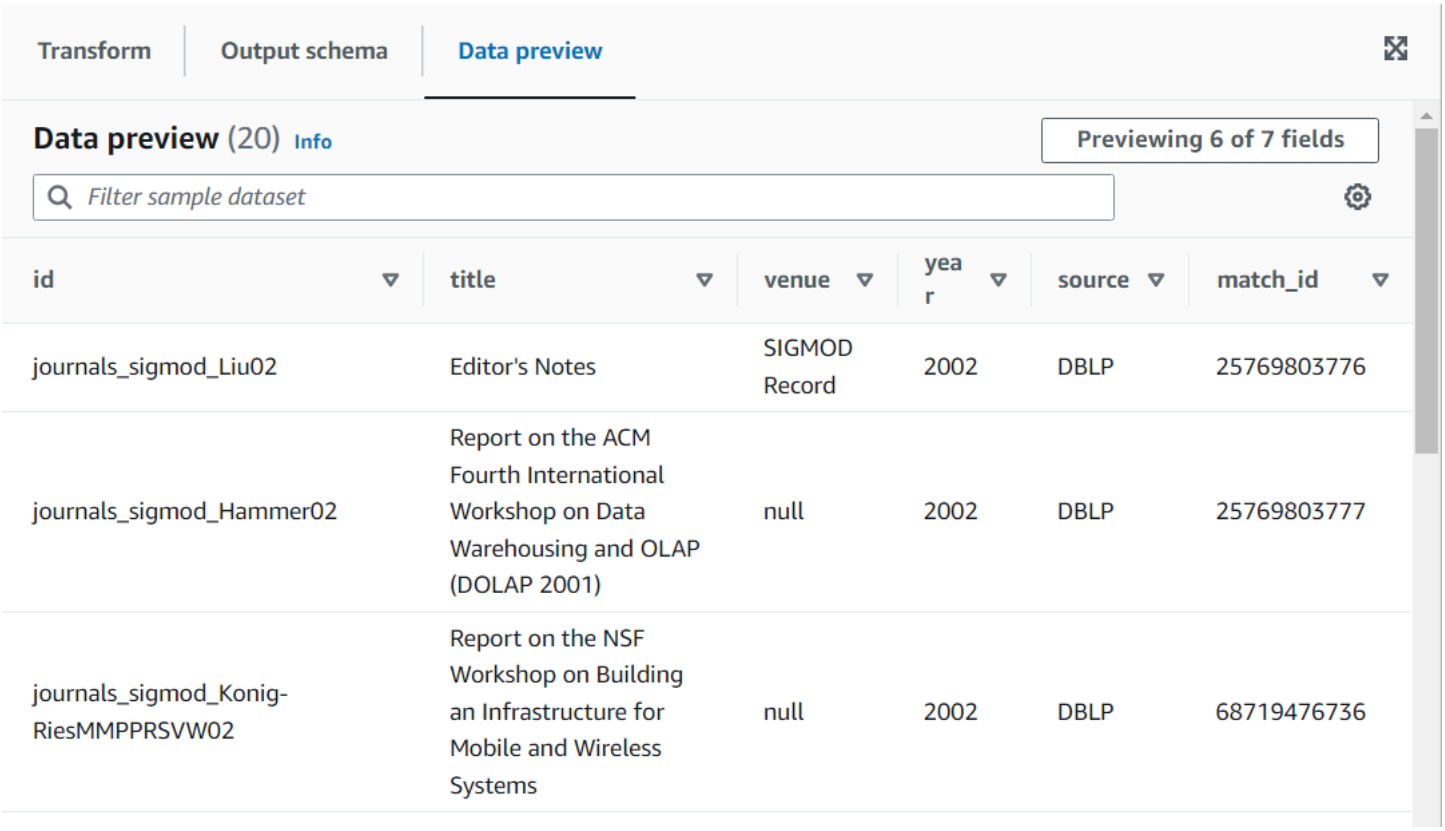
Uso de la transformación de registro de coincidencias para invocar una transformación de clasificación de datos existente

Esta transformación invoca una transformación de clasificación de datos de machine learning de registro de coincidencias existente.

La transformación evalúa los datos actuales y los compara con el modelo entrenado en función de las etiquetas. Se agrega una columna “match_id” para asignar cada fila a un grupo de elementos que

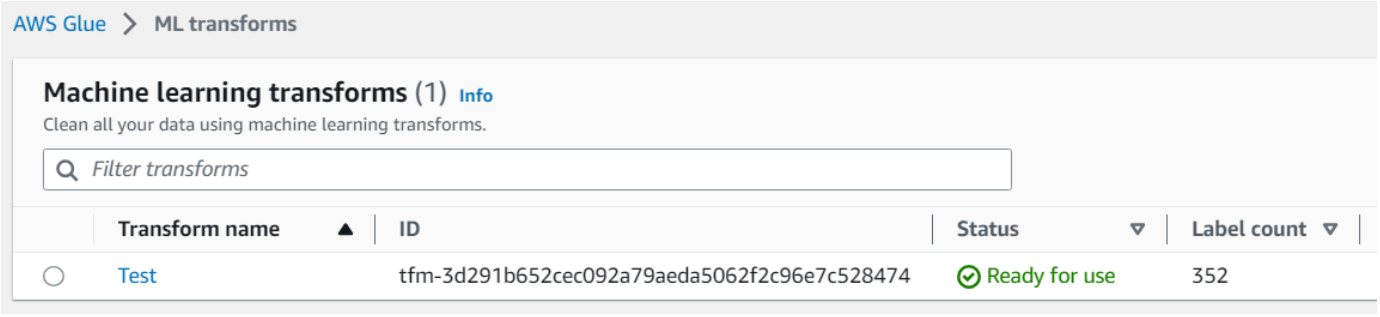
se consideran equivalentes según el entrenamiento del algoritmo. Para obtener más información, consulte Registro de coincidencias con Lake Formation FindMatches.

> **Note**
>
> La versión AWS Glue utilizada en el trabajo visual debe coincidir con la versión que AWS Glue se utilizó para crear la transformación de registro de coincidencias.



Para agregar un nodo de transformación de registro de coincidencias al diagrama de trabajo

1. Abra el panel de recursos y, luego, elija Registro de coincidencias para agregar una nueva transformación al diagrama de trabajo. El nodo seleccionado en el momento de agregar el nodo será el nodo principal.
2. (Opcional) En la pestaña propiedades del nodo, puede ingresar un nombre para el nodo en el diagrama de trabajo. Si todavía no está seleccionado un nodo principal, elija un nodo de la lista Node parents (Nodos principales) para utilizar como origen de entrada para la transformación.
3. En la pestaña Transformación, ingrese el ID obtenido de la página de transformaciones de machine learning:



4. (Opcional) En la pestaña Transformación, se puede marcar la opción para agregar las puntuaciones de confianza. A costa de un cálculo extra, el modelo estimará una puntuación de confianza para cada partido como una columna adicional.

Eliminación de filas nulas

Esta transformación elimina del conjunto de datos las filas que tienen todas las columnas como nulas. Además, se puede ampliar este criterio para incluir campos vacíos, a fin de mantener las filas en las que al menos una columna no esté vacía.

Para agregar un nodo de transformación de eliminación de filas nulas campo al diagrama de trabajo

1. Abra el panel de recursos y, luego, elija Eliminar filas nulas para agregar una nueva transformación al diagrama de trabajo. El nodo seleccionado en el momento de agregar el nodo será el nodo principal.
2. (Opcional) En la pestaña propiedades del nodo, puede ingresar un nombre para el nodo en el diagrama de trabajo. Si todavía no está seleccionado un nodo principal, elija un nodo de la lista Node parents (Nodos principales) para utilizar como origen de entrada para la transformación.
3. (Opcional) En la pestaña Transformación, active la opción Extendida si desea que las filas no solo no sean nulas, sino que tampoco estén vacías, de forma que las cadenas, matrices o mapas vacíos se considerarán nulos a los efectos de esta transformación.

Análisis de una columna de cadena que contiene datos JSON

Esta transformación analiza una columna de cadena que contiene datos JSON y la convierte en una estructura o una columna de matriz, en función de si el JSON es un objeto o una matriz, respectivamente. Si lo desea, puede conservar tanto la columna analizada como la original.

El esquema JSON se puede proporcionar o deducir (en el caso de los objetos de JSON), con un muestreo opcional.

Para agregar un nodo de transformación de columnas JSON de análisis al diagrama de trabajo

1. Abra el panel de recursos y, luego, elija Analizar columna JSON para agregar una nueva transformación al diagrama de trabajo. El nodo seleccionado en el momento de agregar el nodo será el nodo principal.
2. (Opcional) En la pestaña propiedades del nodo, puede ingresar un nombre para el nodo en el diagrama de trabajo. Si todavía no está seleccionado un nodo principal, elija un nodo de la lista Node parents (Nodos principales) para utilizar como origen de entrada para la transformación.
3. En la pestaña Transformación, selecciona la columna que contiene la cadena JSON.
4. (Opcional) En la pestaña Transformación, ingrese el esquema que siguen los datos de JSON con la sintaxis SQL, por ejemplo: “field1 STRING, field2 INT” en el caso de un objeto o “ARRAY<STRING>” en el caso de una matriz.

   Si se trata de una matriz, se requiere el esquema, pero en el caso de un objeto, si no se especifica el esquema, se deducirá a partir de los datos. Para reducir el impacto de inferir el esquema (especialmente en un conjunto de datos grande), puede evitar leer los datos completos dos veces. Para ello, ingrese una proporción de muestras que se utilizará para inferir el esquema. Si el valor es inferior a 1, se utiliza la proporción correspondiente de muestras aleatorias para deducir el esquema. Si los datos son fiables y el objeto es concordante entre las filas, se puede utilizar una proporción pequeña como 0,1 para mejorar el rendimiento.
5. (Opcional) En la pestaña Transformación, se puede ingresar un nombre de columna nuevo si desea conservar tanto la columna de cadena original como la columna analizada.

Extracción de una ruta JSON

Esta transformación extrae nuevas columnas de una columna de cadena JSON. Esta transformación resulta útil cuando solo se necesitan unos pocos elementos de datos y no se desea importar todo el contenido de JSON al esquema de la tabla.

Para agregar un nodo de transformación de Extracción de Ruta JSON al diagrama de trabajo

1. Abra el panel de recursos y, a continuación, elija Extracción de ruta JSON para agregar una nueva transformación al diagrama de trabajo. El nodo seleccionado en el momento de agregar el nodo será el nodo principal.
2. (Opcional) En la pestaña propiedades del nodo, puede ingresar un nombre para el nodo en el diagrama de trabajo. Si todavía no está seleccionado un nodo principal, elija un nodo de la lista Node parents (Nodos principales) para utilizar como origen de entrada para la transformación.

3. En la pestaña Transformación, selecciona la columna que contiene la cadena JSON. Ingrese una o más expresiones de ruta JSON separadas por comas, cada una de las cuales haga referencia a cómo extraer un valor de la matriz o el objeto JSON. Por ejemplo, si la columna JSON contiene un objeto con las propiedades “prop_1” y “prop2”, puede extraer ambos al especificar los nombres “prop_1, prop_2”.

   Si el campo JSON tiene caracteres especiales, por ejemplo, para extraer la propiedad de este JSON `{"a. a": 1}`, puede usar la ruta `$['a. a']`. La excepción es la coma porque está reservada para separar rutas. A continuación, ingrese los nombres de columna correspondientes para cada ruta, separados por comas.

4. (Opcional) En la pestaña Transformación, puedes marcar la casilla para eliminar la columna JSON una vez extraída. Esto tiene sentido si no necesitas el resto de los datos JSON una vez extraídas las partes que necesitas.

Extracción de fragmentos de cadenas mediante una expresión regular

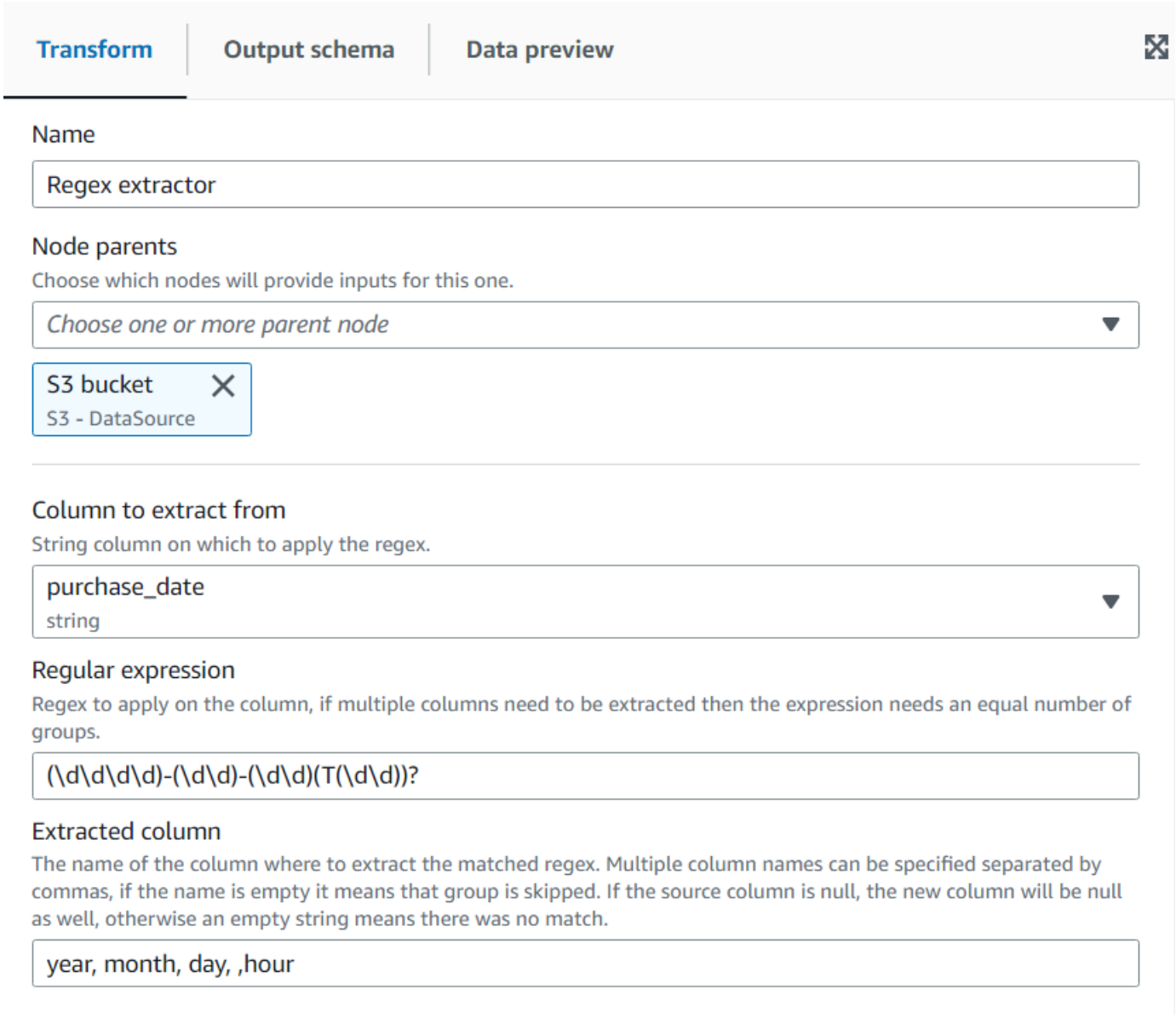
Esta transformación extrae fragmentos de cadenas mediante una expresión regular y crea una nueva columna a partir de ella o varias columnas si se utilizan grupos de expresiones regulares.

Para agregar un nodo de transformación de extractor de expresiones regulares al diagrama de trabajo:

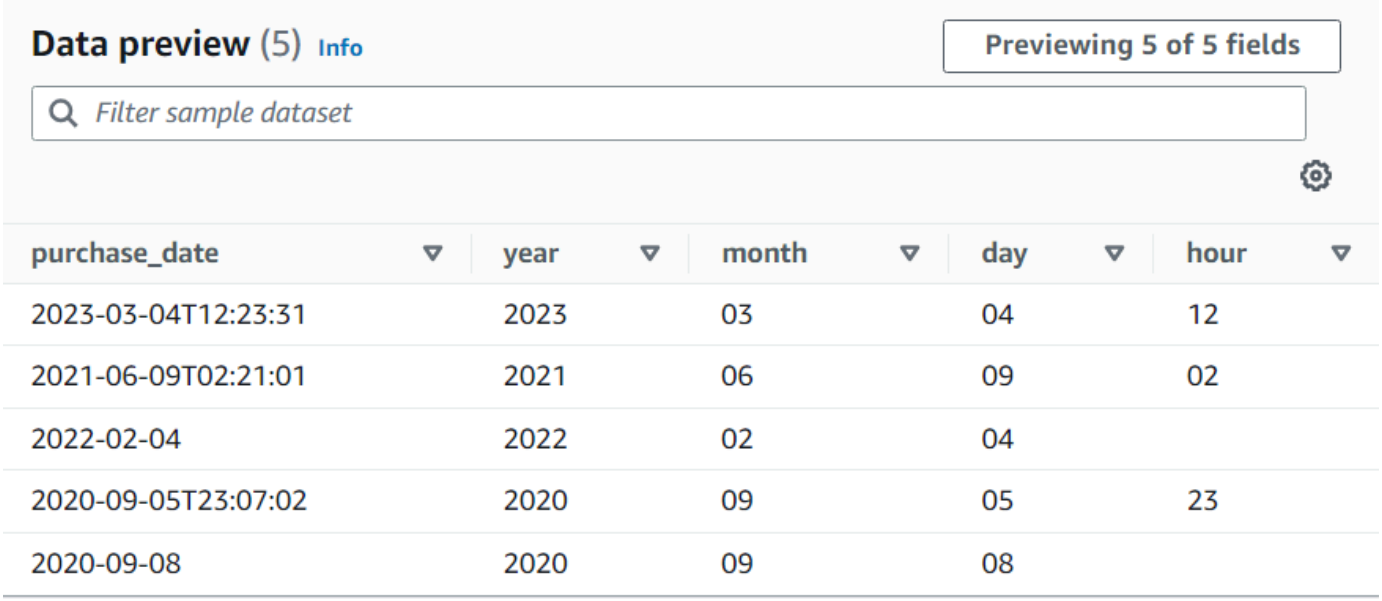
1. Abra el panel de recursos y, luego, elija Extractor de expresiones regulares para agregar una nueva transformación al diagrama de trabajo. El nodo seleccionado en el momento de agregar el nodo será el nodo principal.
2. (Opcional) En la pestaña propiedades del nodo, puede ingresar un nombre para el nodo en el diagrama de trabajo. Si todavía no está seleccionado un nodo principal, elija un nodo de la lista Node parents (Nodos principales) para utilizar como origen de entrada para la transformación.
3. En la pestaña Transformación, ingrese la expresión regular y la columna en la que se debe aplicar. Luego, ingrese el nombre de la nueva columna en la que desee almacenar la cadena correspondiente. La nueva columna será nula solo si la columna de origen es nula; si la expresión regular no coincide, la columna estará vacía.

   Si la expresión regular usa grupos, habrá un nombre de columna correspondiente separado por una coma, pero puede omitir los grupos al dejar el nombre de la columna vacío.

Por ejemplo, si tiene una columna “fecha de compra” con una cadena que utiliza formatos de fecha ISO largos y cortos, querrá extraer el año, el mes, el día y la hora, cuando estén disponibles. Tenga en cuenta que el grupo de horas es opcional; de lo contrario, en las filas donde no esté disponible, todos los grupos extraídos serían cadenas vacías (ya que la expresión regular no coincide). En este caso, no queremos que el grupo haga que la hora sea opcional, sino la interna, por lo que dejamos el nombre en blanco y no se extrae (ese grupo incluiría el carácter T).



Como resultado, se obtiene la vista previa de los datos:



purchase_date	year	month	day	hour
2023-03-04T12:23:31	2023	03	04	12
2021-06-09T02:21:01	2021	06	09	02
2022-02-04	2022	02	04	
2020-09-05T23:07:02	2020	09	05	23
2020-09-08	2020	09	08	

Crear una transformación personalizada

Si necesita realizar transformaciones más complicadas en sus datos o desea agregar claves de propiedad de datos al conjunto de datos, puede agregar una transformación Custom code (Código personalizado) al diagrama de trabajo. El nodo Custom code (Código personalizado) permite introducir un script que realiza la transformación.

Cuando utilice el código personalizado, debe utilizar un editor de esquemas para indicar los cambios realizados en la salida a través del código personalizado. Cuando edita el esquema, puede realizar las siguientes acciones:

- Agregar o eliminar claves de propiedades de datos
- Cambiar el tipo de datos de las claves de propiedad de datos
- Cambiar el nombre de las claves de propiedad de datos
- Reestructurar una clave de propiedad anidada

Debe utilizar una transformación SelectFromCollection (Seleccionar desde la recopilación) para elegir un único `DynamicFrame` del resultado del nodo de transformación personalizado antes de enviar la salida a una ubicación de destino.

Utilice las siguientes tareas para agregar un nodo de transformación personalizado al diagrama de trabajo.

Agregar un nodo de transformación de código personalizado al diagrama de trabajo

Para agregar un nodo de transformación personalizado al diagrama de trabajo

1. (Opcional) abra el panel de recursos y elija Custom transform para agregar una nueva transformación al diagrama de trabajo, si es necesario.
2. En la pestaña Node properties (Propiedades del nodo), ingrese un nombre para el nodo en el diagrama de trabajo. Si todavía no se ha seleccionado un nodo principal, o si desea varias entradas para la transformación personalizada, elija un nodo en la lista Node parents (Nodos principales) que se utilizará como origen de entrada para la transformación.

Introducción de código para el nodo de transformación personalizado

Puede escribir o copiar el código en un campo de entrada. El trabajo utiliza este código para realizar la transformación de datos. Puede proporcionar un fragmento de código en Python o Scala. El código debe tener uno o varios `DynamicFrames` como entrada y devuelve una recopilación de `DynamicFrames`.

Para escribir el script para un nodo de transformación personalizado

1. Con el nodo de transformación personalizado seleccionado en el diagrama de trabajo, elija la pestaña Transform (Transformación).
2. En el campo de entrada de texto en el encabezado Code block (Bloque de código), pegue o escriba el código para la transformación. El código que utilice debe coincidir con el lenguaje especificado para el trabajo en la pestaña Job details (Detalles del trabajo).

   Al hacer referencia a los nodos de entrada en el código, AWS Glue Studio nombra los `DynamicFrames` que devuelven los nodos del diagrama de trabajo de manera secuencial en función del orden de creación. Utilice uno de los siguientes métodos de nomenclatura en el código:

   - Generación de código clásico: utilice nombres funcionales para hacer referencia a los nodos del diagrama de trabajo.
     - Nodo de origen de datos: `DataSource0`, `DataSource1`, `DataSource2`, etc.
     - Nodos de transformación: `Transform0`, `Transform1`, `Transform2`, etc.
   - Nueva generación de código: utilice el nombre especificado en la pestaña Node properties (Propiedades del nodo) de un nodo, anexado con '`_node1`', '`_node2`', y así sucesivamente.

Por ejemplo, `S3bucket_node1`, `ApplyMapping_node2`, `S3bucket_node2`, `MyCustomNodeName_node1`.

Para obtener más información acerca del nuevo generador de código, consulte Generación de código de script.

Los siguientes ejemplos muestran el formato del código que se va a introducir en el cuadro de código:

Python

En el siguiente ejemplo se toma el primer `DynamicFrame` recibido, se convierte en un valor de `DataFrame` para aplicar el método de filtro nativo (se mantienen solo los registros que tienen más de 1000 votos), luego se convierte nuevamente en un `DynamicFrame` antes de que se devuelva.

```
def FilterHighVoteCounts (glueContext, dfc) -> DynamicFrameCollection:
    df = dfc.select(list(dfc.keys())[0]).toDF()
    df_filtered = df.filter(df["vote_count"] > 1000)
    dyf_filtered = DynamicFrame.fromDF(df_filtered, glueContext, "filter_votes")
    return(DynamicFrameCollection({"CustomTransform0": dyf_filtered}, glueContext))
```

Scala

En el siguiente ejemplo se toma el primer `DynamicFrame` recibido, se convierte en un valor de `DataFrame` para aplicar el método de filtro nativo (se mantienen solo los registros que tienen más de 1000 votos), luego se convierte nuevamente en un `DynamicFrame` antes de que se devuelva.

```
object FilterHighVoteCounts {
  def execute(glueContext : GlueContext, input : Seq[DynamicFrame]) :
 Seq[DynamicFrame] = {
    val frame = input(0).toDF()
    val filtered = DynamicFrame(frame.filter(frame("vote_count") > 1000),
 glueContext)
    Seq(filtered)
  }
}
```

Edición de esquema para un nodo de transformación personalizado

Cuando utiliza un nodo de transformación personalizado, AWS Glue Studio no puede inferir automáticamente los esquemas de salida creados por la transformación. Utilice el editor de esquemas para describir los cambios de esquema implementados por el código de transformación personalizado.

Un nodo de código personalizado puede tener cualquier número de nodos principales, cada uno de los cuales proporciona un `DynamicFrame` como entrada para su código personalizado. Un nodo de código personalizado devuelve una recopilación de `DynamicFrames`. Cada `DynamicFrame` que se utiliza como entrada tiene asociado un esquema. Debe agregar un esquema que describa cada `DynamicFrame` devuelto por el nodo de código personalizado.

> **Note**
>
> Cuando configura su propio esquema en una transformación personalizada, AWS Glue Studio no hereda esquemas de nodos anteriors. Para actualizar el esquema, seleccione el nodo de transformación personalizada y, a continuación, elija la pestaña Data preview (Vista previa de datos). Una vez generada la vista previa, elija 'Use Preview Schema' (Usar esquema de vista previa). A continuación, el esquema será reemplazado por el esquema utilizando los datos de vista previa.

Para editar los esquemas de un nodo de transformación personalizado

1. Con el nodo de transformación personalizado seleccionado en el diagrama de trabajo, elija la pestaña Output schema (Esquema de salida), en el panel de detalles del nodo.
2. Seleccione Edit (Editar) para realizar cambios al esquema.

   Si tiene claves de propiedad de datos anidadas, como una matriz u objeto, puede elegir el ícono Expand-Rows (Expandir filas) () en la parte superior derecha del panel de cada esquema para expandir la lista de claves de propiedades de datos secundarias. Después de seleccionarlo, el ícono cambia a Collapse-Rows (Contraer filas) (), que puede elegir para contraer la lista de claves de propiedad secundarias.

3. Modifique el esquema mediante las siguientes acciones en la sección situada en la parte derecha de la página:

   - Para cambiar el nombre de una clave de propiedad, coloque el cursor en el cuadro de texto Key (Clave) para la clave de propiedad y, a continuación, escriba el nuevo nombre.
   - Para cambiar el tipo de datos de una clave de propiedad, utilice la lista para elegir un nuevo tipo de datos para la clave de propiedad.
   - Para agregar una nueva clave de propiedad de nivel superior al esquema, elija el ícono Overflow (Desbordamiento) (⋯) a la izquierda del botón Cancel (Cancelar) y luego elija Add root key (Agregar clave raíz).
   - Para agregar una clave de propiedad secundaria al esquema, elija el ícono Add-Key (Agrega clave) asociado a la clave principal. Escriba un nombre para la clave secundaria y elija el tipo de datos.
   - Para eliminar una clave de propiedad del esquema, elija el ícono Remove (Eliminar) (🗑) en el extremo derecho del nombre de la clave.

4. Si su código de transformación personalizado utiliza múltiples `DynamicFrames`, puede agregar esquemas de salida adicionales.

   - Para agregar un esquema nuevo, vacío, elija la opción Overflow (Desbordamiento) (⋯) y, a continuación, elija Add output schema (Agregar esquema de salida).
   - Para copiar un esquema existente en un nuevo esquema de salida, asegúrese de que el esquema que desea copiar se muestra en el selector de esquema. Elija el ícono Overflow (Desbordamiento) (⋯) y, a continuación, elija Duplicate (Duplicar).

   Si desea eliminar un esquema de salida, asegúrese de que el esquema que desea copiar se muestra en el selector de esquema. Elija el ícono Overflow (Desbordamiento) (⋯) y, a continuación, elija Delete (Eliminar).

5. Agregue nuevas claves raíz al nuevo esquema o edite las claves duplicadas.
6. Cuando modifique los esquemas de salida, elija el botón Apply (Aplicar) para guardar los cambios y salir del editor de esquemas.

   Si no desea guardar los cambios, elija Cancel (Cancelar).

Configurar la salida de transformación personalizada

Una transformación de código personalizado devuelve una recopilación de `DynamicFrames`, aún cuando haya solo un `DynamicFrame` en el conjunto de resultados.

Para procesar la salida desde un nodo de transformación personalizado

1. Agregue un nodo de transformación SelectFromCollection (Seleccionar desde la recopilación), que tenga el nodo de transformación personalizado como su nodo principal. Actualice esta transformación para indicar qué conjunto de datos desea utilizar. Para obtener más información, consulte Uso de SelectFromCollection (Seleccionar desde la recopilación) para elegir qué conjunto de datos desea mantener.
2. Agregue una transformación Seleccionar desde recopilación adicional al diagrama de trabajo si desea utilizar `DynamicFrames` adicionales, producidos por el nodo de transformación personalizado.

   Considere un escenario en el que agrega un nodo de transformación personalizado para dividir un conjunto de datos de vuelo en varios conjuntos de datos, pero duplica algunas de las claves de propiedad de identificación en cada esquema de salida, como la fecha de vuelo o el número de vuelo. Agregue un nodo de transformación Seleccionar desde la recopilación para cada esquema de salida, con el nodo de transformación personalizado como su nodo principal.
3. (Opcional) a continuación, puede usar cada nodo de transformación Seleccionar desde la recopilación como entrada para otros nodos del trabajo, o como nodo principal para un nodo de destino de datos.

Transformaciones visuales personalizadas de AWS Glue

Las transformaciones visuales personalizadas permiten crear transformaciones y ponerlas a disposición de los usuarios para que las utilicen en trabajos de AWS Glue Studio. Las transformaciones visuales personalizadas permiten a los desarrolladores de ETL, que tal vez no

estén familiarizados con la codificación, buscar y utilizar una biblioteca de transformaciones cada vez mayor mediante la interfaz de AWS Glue Studio.

Puede crear una transformación visual personalizada y, a continuación, subirla a Amazon S3 para que esté disponible para su uso a través del editor visual en AWS Glue Studio para ocuparse de estos trabajos.

Temas

- Introducción a las transformaciones visuales personalizadas
- Paso 1. Crear un archivo de configuración JSON
- Paso 2. Implementar la lógica de transformación
- Paso 3. Validar y solucionar los problemas de las transformaciones visuales personalizadas en AWS Glue Studio
- Paso 4. Actualizar las transformaciones visuales personalizadas según sea necesario
- Paso 5. Usar transformaciones visuales personalizadas en AWS Glue Studio
- Ejemplos de uso
- Ejemplos de scripts visuales personalizados
- Vídeo

Introducción a las transformaciones visuales personalizadas

Para crear una transformación visual personalizada, siga estos pasos.

- Paso 1. Crear un archivo de configuración JSON
- Paso 2. Implementar la lógica de transformación
- Paso 3. Validar la transformación visual personalizada
- Paso 4. Actualizar la transformación visual personalizada según sea necesario
- Paso 5. Utilizar la transformación visual personalizada en AWS Glue Studio

Para empezar, configure el bucket de Amazon S3 y continúe con el Paso 1. Crear un archivo de configuración JSON.

Requisitos previos

Las transformaciones proporcionadas por los clientes residen en una cuenta de AWS de cliente. Esa cuenta es la propietaria de las transformaciones y, por lo tanto, tiene todos los permisos para verlas (buscarlas y usarlas), editarlas o eliminarlas.

Para utilizar una transformación personalizada en AWS Glue Studio, tendrá que crear y subir dos archivos al bucket de activos de Amazon S3 de esa cuenta de AWS:

- Archivo Python: contiene la función de transformación
- Archivo JSON: describe la transformación. También se conoce como el archivo de configuración que se requiere para definir la transformación.

Para emparejar los archivos, utilice el mismo nombre para ambos. Por ejemplo:

- myTransform.json
- myTransform.py

Si lo desea, puede dar a una transformación visual personalizada un icono personalizado al proporcionar un archivo SVG que contenga el icono. Para emparejar los archivos, utilice el mismo nombre para el icono:

- myTransform.svg

AWS Glue Studio los combinará automáticamente mediante los nombres de archivo correspondientes. Los nombres de los archivos no pueden ser los mismos para los módulos existentes.

Convención recomendada para el nombre del archivo de transformación

AWS Glue Studio importará el archivo como un módulo (por ejemplo, `import myTransform`) en el script del trabajo. Por lo tanto, el nombre del archivo debe seguir las mismas reglas de nomenclatura establecidas para los nombres de variables de Python (identificadores). En concreto, deben empezar por una letra o un guion bajo y estar compuestos en su totalidad por letras, dígitos o guiones bajos.

> **Note**
>
> Asegúrese de que el nombre del archivo de transformación no entre en conflicto con los módulos de Python cargados existentes (por ejemplo, `sys, array, copy`, etc.) para evitar problemas inesperados en tiempo de ejecución.

Configuración del bucket de Amazon S3

Las transformaciones que cree se almacenan en Amazon S3 y son propiedad de su cuenta de AWS. Para crear nuevas transformaciones visuales personalizadas, basta con que suba archivos (json y py) a la carpeta de activos de Amazon S3, en la que se almacenan actualmente los scripts de todos los trabajos (por ejemplo, `s3://aws-glue-assets-<accountid>-<region>/transforms`). Si utiliza un icono personalizado, cárguelo también. De forma predeterminada, AWS Glue Studio leerá todos los archivos. json de la carpeta /transforms en el mismo bucket de S3.

Paso 1. Crear un archivo de configuración JSON

Se necesita un archivo de configuración JSON para definir y describir la transformación visual personalizada. A continuación se indica el esquema para el archivo de configuración.

Estructura del archivo JSON

Campos

- `name: string`: (obligatorio) nombre del sistema de transformación utilizado para identificar las transformaciones. Siga las mismas reglas de nomenclatura establecidas para los nombres de variables de Python (identificadores). En concreto, deben empezar por una letra o un guion bajo y estar compuestos en su totalidad por letras, dígitos o guiones bajos.
- `displayName: string`: (opcional) nombre de la transformación que se muestra en el editor de trabajos visuales de AWS Glue Studio. Si no se especifica ningún valor de `displayName`, el valor de `name` se usa como nombre de la transformación en AWS Glue Studio.
- `description: string`: (opcional) la descripción de la transformación se muestra en AWS Glue Studio y se puede buscar.
- `functionName: string`: (obligatorio) el nombre de la función de Python se utiliza para identificar la función que se va a invocar en el script de Python.
- `path: string`: (opcional) la ruta completa de Amazon S3 al archivo de origen de Python. Si no se especifica, AWS Glue utiliza la coincidencia de nombres de archivo para emparejar

los archivos .json y .py. Por ejemplo, el nombre del archivo JSON (`myTransform.json`) se emparejará con el archivo Python (`myTransform.py`) en la misma ubicación de Amazon S3.

- `parameters: Array of TransformParameter object`: (opcional) la lista de parámetros que se va a mostrar al configurarlos en el editor visual de AWS Glue Studio.

Campos de TransformParameters

- `name: string`: (obligatorio) el nombre del parámetro que se pasará a la función de Python como argumento con nombre en el script del trabajo. Siga las mismas reglas de nomenclatura establecidas para los nombres de variables de Python (identificadores). En concreto, deben empezar por una letra o un guion bajo y estar compuestos en su totalidad por letras, dígitos o guiones bajos.
- `displayName: string`: (opcional) nombre de la transformación que se muestra en el editor de trabajos visuales de AWS Glue Studio. Si no se especifica ningún valor de `displayName`, el valor de `name` se usa como nombre de la transformación en AWS Glue Studio.
- `type: string`: (obligatorio) el tipo de parámetro que acepta los tipos de datos comunes de Python. Valores válidos: “str” | “int” | “float” | “list” | “bool”.
- `isOptional: boolean`: (opcional) determina si el parámetro es opcional. De forma predeterminada, todos los parámetros son obligatorios.
- `description: string`: (opcional) la descripción se muestra en AWS Glue Studio para ayudar al usuario a configurar el parámetro de transformación.
- `validationType: string`: (opcional) define la forma en que se valida este parámetro. Actualmente, solo admite expresiones regulares. De forma predeterminada, el tipo de validación se establece en `RegularExpression`.
- `validationRule: string`: (opcional) expresión regular que se utiliza para validar la entrada del formulario antes de enviarlo cuando `validationType` se establece en `RegularExpression`. La sintaxis de las expresiones regulares debe ser compatible con las especificaciones de RegExp de ECMAScript.
- `validationMessage: string`: (opcional) el mensaje que se mostrará cuando se produzca un error en la validación.
- `listOptions: An array of TransformParameterListOption object` O un `string` o el valor de cadena 'columna': (opcional) opciones para mostrar en el control de UI Seleccionar o Selección múltiple. Acepta una lista de valores separados por comas o un objeto JSON fuertemente tipado de tipo `TransformParameterListOption`. También puede rellenar

dinámicamente la lista de columnas del esquema del nodo principal al especificar el valor de cadena “column”.

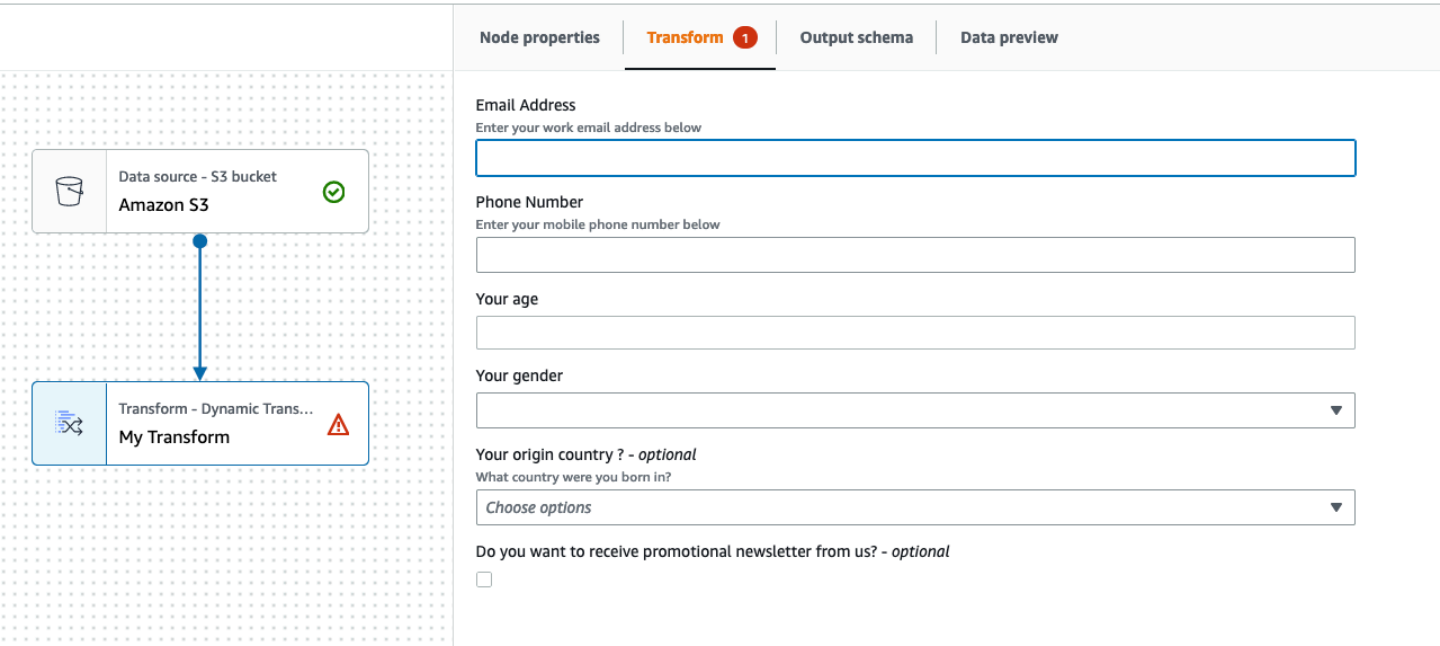
- `listType: string`: (opcional) se definen los tipos de opciones para type = “list”. Valores válidos: “str” | “int” | “float” | “list” | “bool”. Tipo de parámetro que acepta los tipos de datos comunes de Python.

Campos de TransformParameterListOption

- `value: string | int | float | bool`: (obligatorio) valor de la opción.
- `label: string`: (opcional) etiqueta de la opción que se muestra en el menú desplegable de selección.

Parámetros de transformación en AWS Glue Studio

De forma predeterminada, los parámetros son obligatorios a menos que se marquen como `isOptional` en el archivo .json. En AWS Glue Studio, los parámetros se muestran en la pestaña Transform (Transformación). En el ejemplo se muestran parámetros definidos por el usuario, como Email Address (Dirección de correo electrónico), Phone Number (Número de teléfono), Your age (Su edad), Your gender (Su sexo) y Your origin country (Su país de origen).

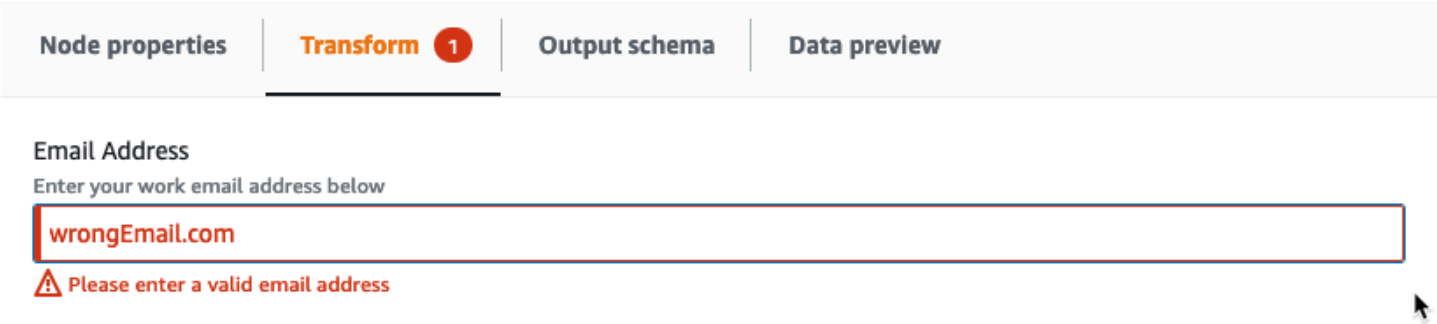


Puede aplicar algunas validaciones en AWS Glue Studio mediante expresiones regulares en el archivo json; para ello, especifique el parámetro `validationRule` y un mensaje de validación en `validationMessage`.

```
    "validationRule": "^\\(?(\\d{3})\\)?[- ]?(\\d{3})[- ]?(\\d{4})$",
    "validationMessage": "Please enter a valid US number"
```

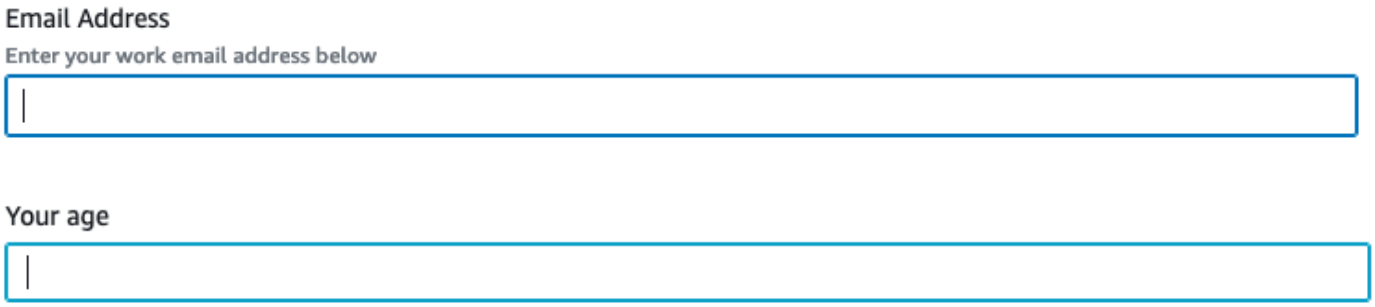
> **Note**
>
> Como la validación se lleva a cabo en el navegador, la sintaxis de la expresión regular debe ser compatible con las especificaciones de RegExp de ECMAScript. La sintaxis de Python no es compatible con estas expresiones regulares.

Al agregar la validación, se impedirá al usuario guardar el trabajo con una entrada de usuario incorrecta. AWS Glue Studio muestra el mensaje de validación, tal como se muestra en el ejemplo:

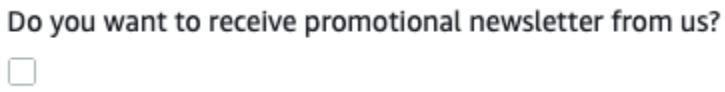


Los parámetros se muestran en AWS Glue Studio en función de la configuración de los parámetros.

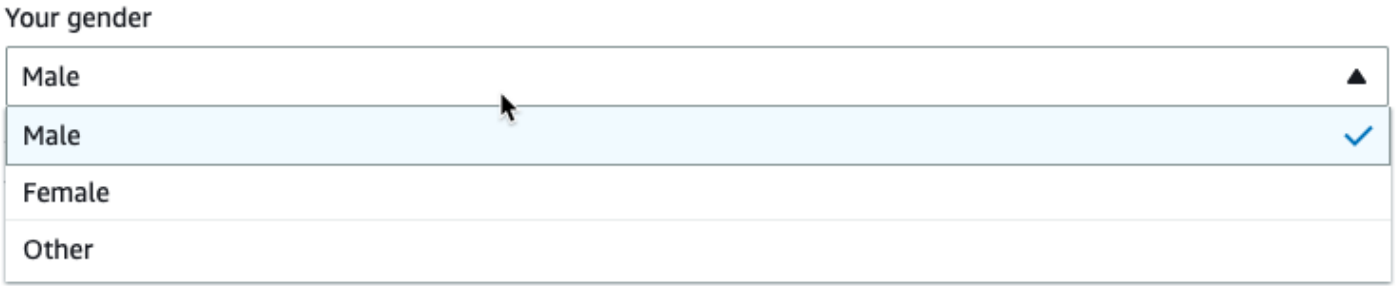
- Se muestra un campo de entrada de texto cuando el valor de `type` es uno de los siguientes: `str`, `int` o `float`. Por ejemplo, en la captura de pantalla se muestran los campos de entrada para los parámetros “Dirección de correo electrónico” y “Su edad”.



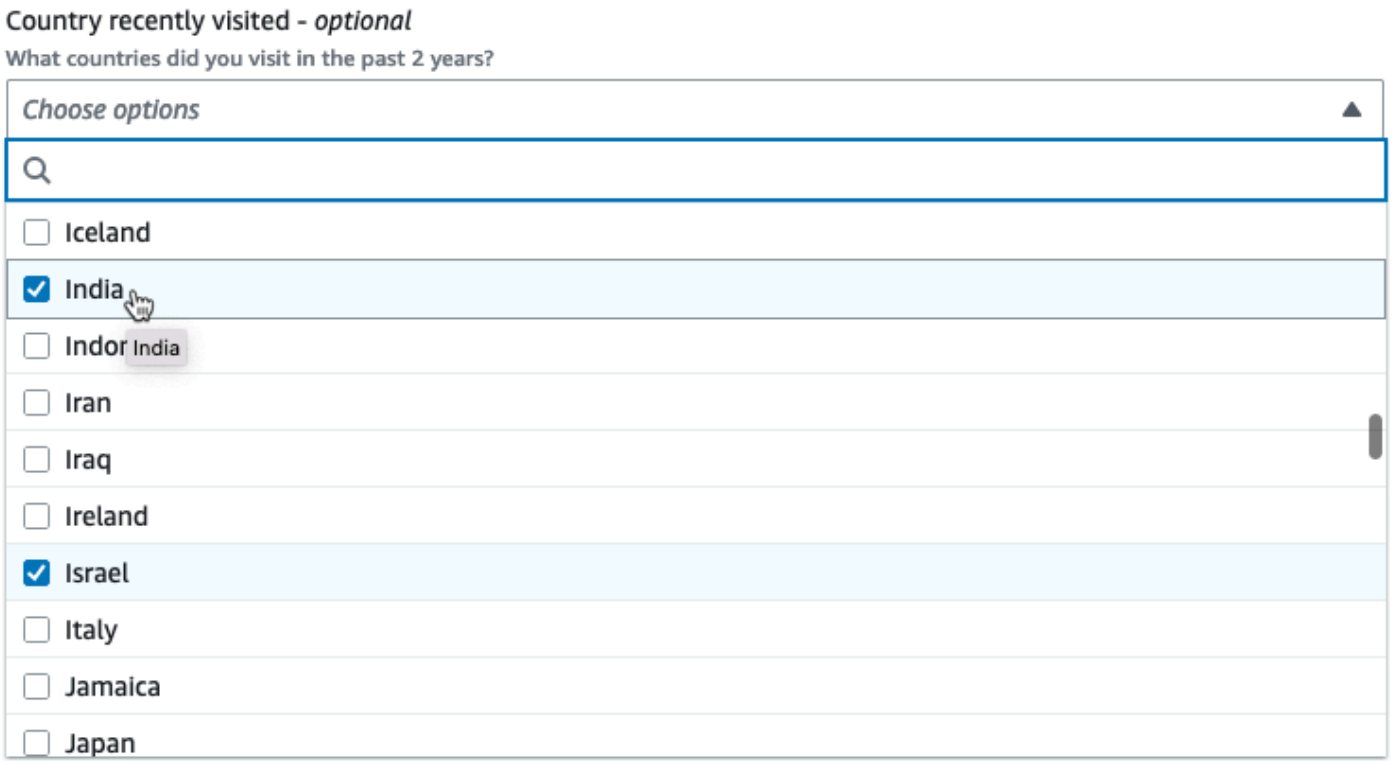
- Cuando el valor de `type` es `bool`, se muestra una casilla de verificación.



- Cuando el valor de `type` es `str` y se proporciona `listOptions`, se muestra una lista de selección única.



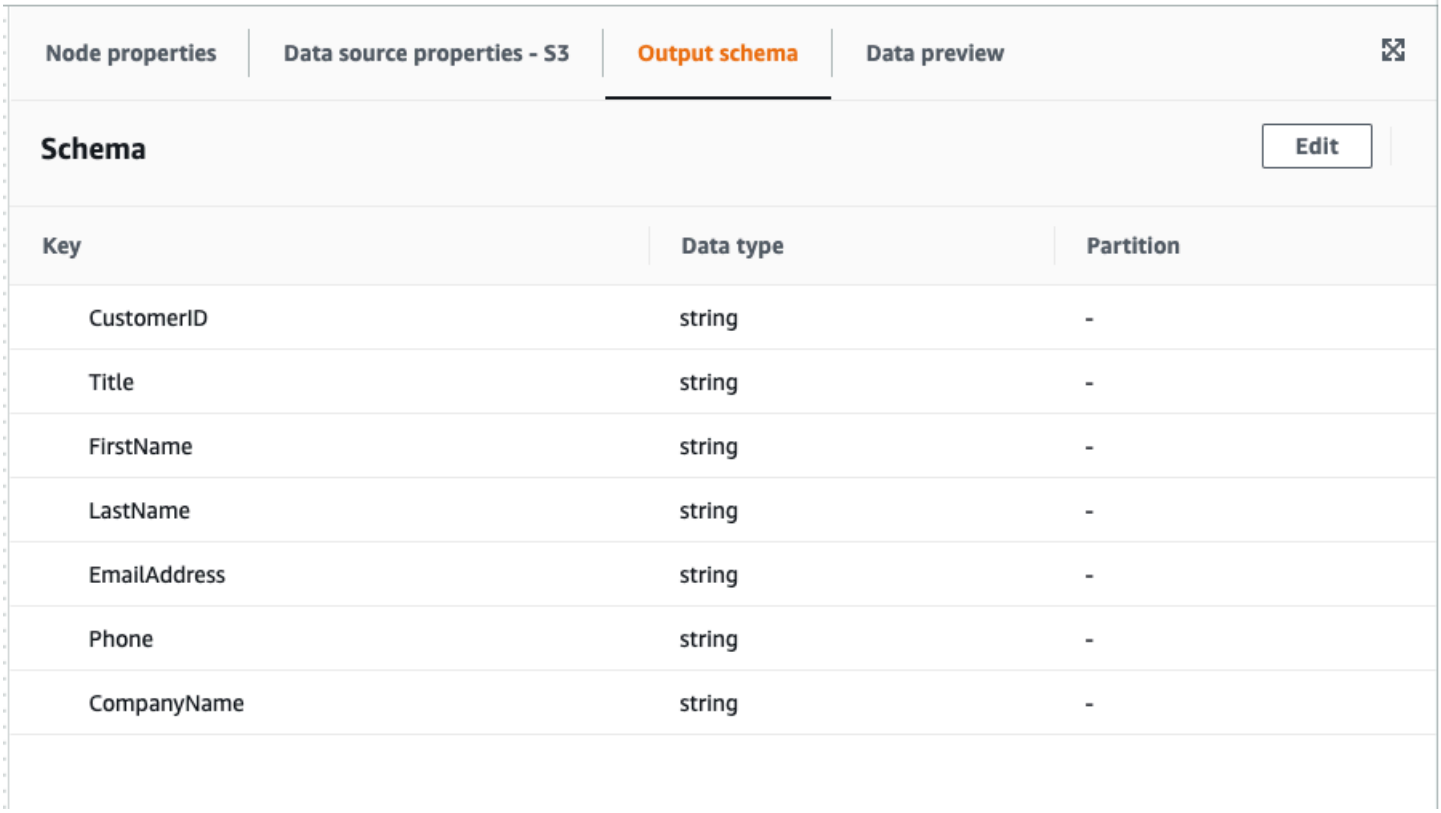
- Cuando el valor de `type` es `list` y se proporcionan `listOptions` y `listType`, se muestra una lista de selección múltiple.



Mostrar un selector de columnas como parámetro

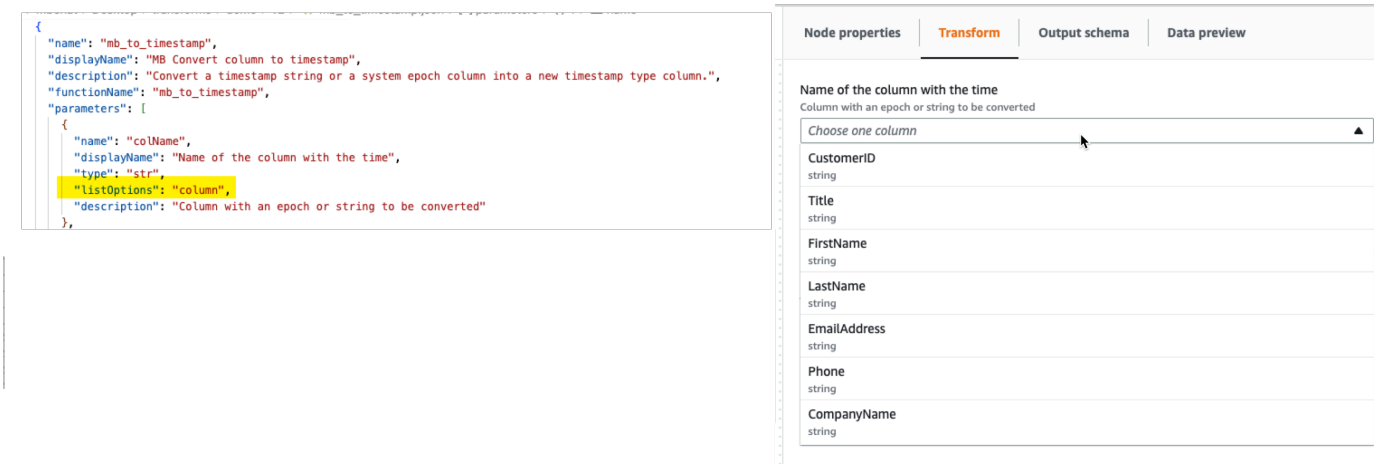
Si la configuración requiere que el usuario elija una columna del esquema, puede mostrar un selector de columnas para que el usuario no tenga que escribir el nombre de la columna. Al configurar el campo `listOptions` como “columna”, AWS Glue Studio muestra dinámicamente un selector de columnas basado en el esquema de salida del nodo principal. AWS Glue Studio puede mostrar un selector de una o varias columnas.

En este ejemplo se usa el esquema:



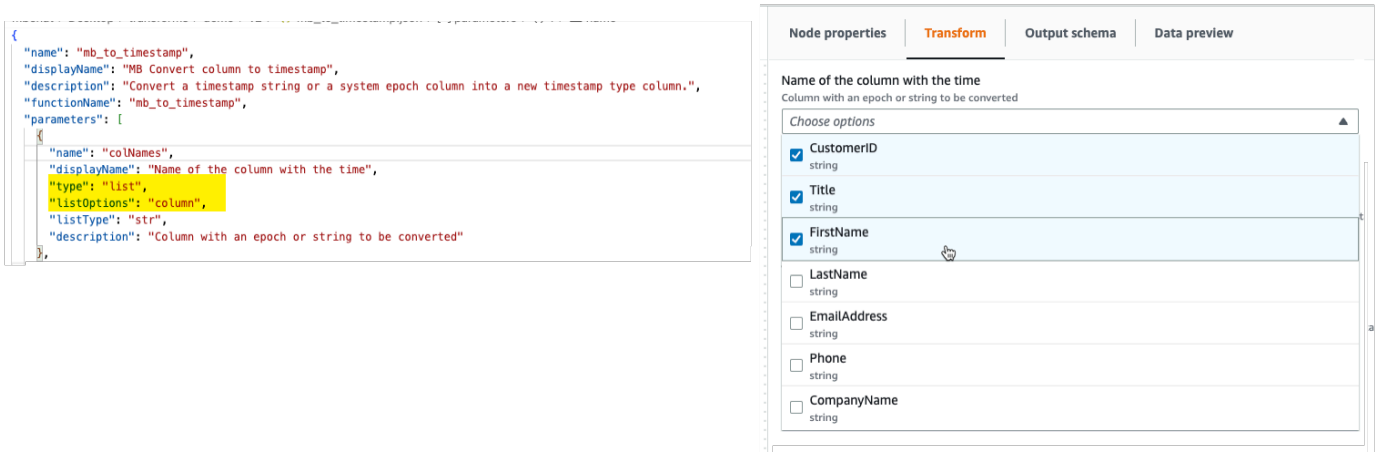
Para definir el parámetro de transformación visual personalizada para que muestre una sola columna:

1. En el archivo JSON, para el objeto `parameters`, establece el valor `listOptions` en “columna”. Esto permite al usuario elegir una columna de una lista de selección de AWS Glue Studio.



2. También puede permitir la selección de varias columnas al definir el parámetro de la siguiente manera:

   - `listOptions: "column"`
   - `type: "list"`



Paso 2. Implementar la lógica de transformación

> **Note**
>
> Las transformaciones visuales personalizadas solo admiten scripts de Python. No se admite Scala.

Para agregar el código que implementa la función definida mediante el archivo de configuración .json, se recomienda colocar el archivo de Python en la misma ubicación que el archivo .json, con el mismo nombre, pero con la extensión ".py". AWS Glue Studio empareja automáticamente los archivos .json y .py para que no tenga que especificar la ruta del archivo Python en el archivo de configuración.

En el archivo de Python, agregue la función declarada, con los parámetros con nombre configurados y regístrela para que se pueda utilizar en `DynamicFrame`. A continuación, se muestra un ejemplo de un archivo de Python:

```
from awsglue import DynamicFrame

# self refers to the DynamicFrame to transform,
# the parameter names must match the ones defined in the config
# if it's optional, need to provide a default value
def myTransform(self, email, phone, age=None, gender="",
                      country="", promotion=False):
   resulting_dynf = # do some transformation on self
   return resulting_dynf

DynamicFrame.myTransform = myTransform
```

Se recomienda utilizar un cuaderno de AWS Glue para desarrollar y probar el código de Python de la forma más rápida. Consulte Introducción a los cuadernos en AWS Glue Studio

Para ilustrar cómo implementar la lógica de transformación, la transformación visual personalizada del ejemplo siguiente es una transformación para filtrar los datos entrantes y conservar solo los datos relacionados con un estado concreto de EE. UU. El archivo .json contiene el parámetro para `functionName` como `custom_filter_state` y dos argumentos ("state" y "colName" con el tipo "str").

El archivo .json de configuración de ejemplo es el siguiente:

```
{
"name": "custom_filter_state",
"displayName": "Filter State",
"description": "A simple example to filter the data to keep only the state indicated.",
"functionName": "custom_filter_state",
"parameters": [
   {
```

```
    "name": "colName",
    "displayName": "Column name",
    "type": "str",
    "description": "Name of the column in the data that holds the state postal code"
   },
   {
    "name": "state",
    "displayName": "State postal code",
    "type": "str",
    "description": "The postal code of the state whole rows to keep"
   }
  ]
}
```

Implementación del script complementario en Python

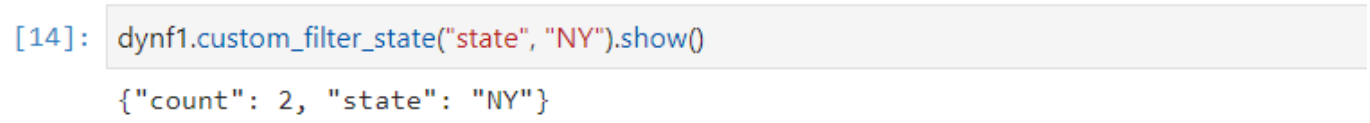
1. Inicie un cuaderno de AWS Glue y ejecute la celda inicial proporcionada para iniciar la sesión. Al ejecutar la celda inicial, se crean los componentes básicos necesarios.
2. Cree una función que filtre tal y como se describe en el ejemplo y regístrela en `DynamicFrame`. Copie el código siguiente y péguelo en una celda del cuaderno de AWS Glue.

```
from awsglue import DynamicFrame

def custom_filter_state(self, colName, state):
    return self.filter(lambda row: row[colName] == state)

DynamicFrame.custom_filter_state = custom_filter_state
```

3. Cree o cargue datos de muestra para probar el código en la misma celda o en una nueva. Si agrega los datos de muestra en una celda nueva, no olvide ejecutar la celda. Por ejemplo:

```
# A few of rows of sample data to test
data_sample = [
    {"state": "CA", "count": 4},
    {"state": "NY", "count": 2},
    {"state": "WA", "count": 3}
]
df1 = glueContext.sparkSession.sparkContext.parallelize(data_sample).toDF()
dynf1 = DynamicFrame.fromDF(df1, glueContext, None)
```

4. Haga pruebas para validar “custom_filter_state” con distintos argumentos:



5. Tras ejecutar varias pruebas, guarde el código con la extensión .py y asigne un nombre al archivo .py que refleje el nombre del archivo .json. Los archivos .py y .json deben estar en la carpeta de la misma transformación.

   Copie el siguiente código, péguelo en un archivo y cámbiele el nombre con una extensión de archivo .py.

```
from awsglue import DynamicFrame

def custom_filter_state(self, colName, state):
    return self.filter(lambda row: row[colName] == state)

DynamicFrame.custom_filter_state = custom_filter_state
```

6. En AWS Glue Studio, abra un trabajo visual y agregue la transformación al trabajo; para ello, selecciónela en la lista Transforms (Transformaciones) disponibles.

   Para reutilizar esta transformación en el código de un script de Python, agregue la ruta de Amazon S3 al archivo .py en el trabajo, en “Referenced files path” (Ruta de archivos a la que se hace referencia), y en el script. Por último, importe el nombre del archivo de Python (sin la extensión); para ello, agréguelo a la parte superior del archivo. Por ejemplo: `import` <nombre del archivo (sin la extensión)>

Paso 3. Validar y solucionar los problemas de las transformaciones visuales personalizadas en AWS Glue Studio

AWS Glue Studio valida el archivo de configuración JSON antes de cargar las transformaciones visuales personalizadas en AWS Glue Studio. La validación incluye lo siguiente:

- Presencia de campos obligatorios

- Validación de formato JSON
- Parámetros incorrectos o no válidos
- Presencia de los archivos .py y .json en la misma ruta de Amazon S3
- Coincidencia de los nombres de los archivos .py y .json

Si la validación se lleva a cabo correctamente, la transformación aparece en la lista Actions (Acciones) disponibles del editor visual. Si se ha proporcionado un icono personalizado, debería estar visible junto a la Acción.

Si se produce un error durante la validación, AWS Glue Studio no carga la transformación visual personalizada.

Paso 4. Actualizar las transformaciones visuales personalizadas según sea necesario

Una vez creado y utilizado, el script de la transformación se puede actualizar siempre que la transformación siga la definición JSON correspondiente:

- El nombre utilizado durante la asignación a DynamicFrame debe coincidir con el valor de `functionName` de JSON.
- Los argumentos de la función deben definirse en el archivo JSON tal y como se describe en Paso 1. Crear un archivo de configuración JSON .
- La ruta de Amazon S3 del archivo de Python no puede cambiar, ya que los trabajos dependen directamente de ella.

> **Note**
>
> Si es necesario llevara a cabo alguna actualización, asegúrese de que el script y el archivo .json se actualicen de forma coherente y que todos los trabajos visuales se vuelvan a guardar correctamente con la nueva transformación. Si los trabajos visuales no se guardan después de llevar a cabo las actualizaciones, estas no se aplicarán ni validarán. Si se cambia el nombre del archivo de script de Python o no se coloca junto al archivo .json, debe especificar la ruta completa en el archivo .json.

Icono personalizado

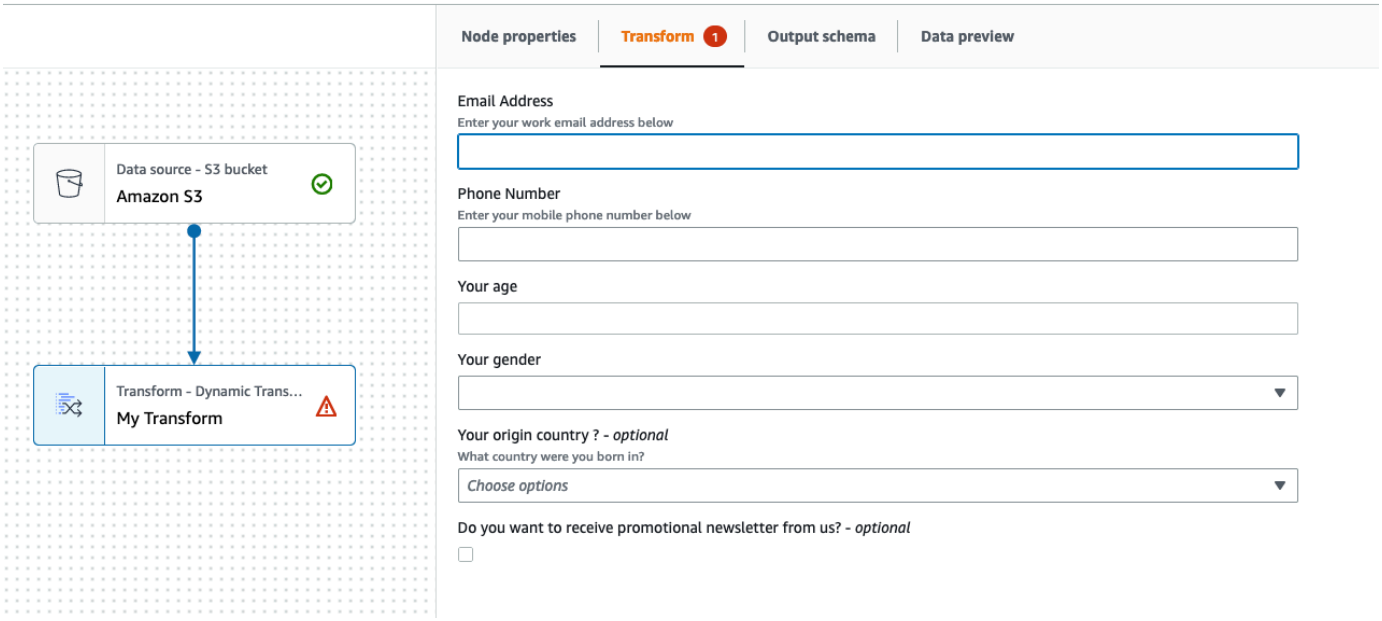
Si determina que el icono predeterminado de la acción no la distingue visualmente como parte de sus flujos de trabajo, puede proporcionar un icono personalizado, como se describe en the section called " Introducción a las transformaciones visuales personalizadas ". Puede actualizar el icono mediante la actualización del SVG correspondiente alojado en Amazon S3.

Para obtener mejores resultados, diseñe la imagen para que se vea a 32 x 32 píxeles según las pautas del sistema de diseño de Cloudscape. Para obtener más información sobre las pautas de Cloudscape, consulte la documentación de Cloudscape

Paso 5. Usar transformaciones visuales personalizadas en AWS Glue Studio

Para utilizar una transformación visual personalizada en AWS Glue Studio, suba los archivos de origen y configuración y, a continuación, seleccione la transformación en el menú Action (Acción). Todos los parámetros que necesiten valores o entrada están disponibles en la pestaña Transform (Transformación).

1. Suba los dos archivos (el archivo de origen de Python y el archivo de configuración JSON) a la carpeta de activos de Amazon S3 en la que se almacenan los scripts del trabajo. De forma predeterminada, AWS Glue extrae todos los archivos JSON de la carpeta /transforms del mismo bucket de Amazon S3.
2. En el menú Action (Acción), seleccione la transformación visual personalizada. Se nombra con el valor de `displayName` de la transformación o el nombre que especificó en el archivo de configuración .json.
3. Ingrese valores para los parámetros configurados en el archivo de configuración.



Ejemplos de uso

A continuación se incluye es un ejemplo de todos los parámetros posibles en un archivo de configuración .json.

```
{
  "name": "MyTransform",
  "displayName": "My Transform",
  "description": "This transform description will be displayed in UI",
  "functionName": "myTransform",
  "parameters": [
      {
      "name": "email",
      "displayName": "Email Address",
      "type": "str",
      "description": "Enter your work email address below",
      "validationType": "RegularExpression",
      "validationRule": "^\\w+([\\.-]?\\w+)*@\\w+([\\.-]?\\w+)*(\\.\\w{2,3})+$",
      "validationMessage": "Please enter a valid email address"
    },
    {
      "name": "phone",
      "displayName": "Phone Number",
      "type": "str",
      "description": "Enter your mobile phone number below",
      "validationRule": "^\\(?(\\d{3})\\)?[- ]?(\\d{3})[- ]?(\\d{4})$",
      "validationMessage": "Please enter a valid US number"
    },
    {
      "name": "age",
      "displayName": "Your age",
      "type": "int",
      "isOptional": true
    },
    {
      "name": "gender",
      "displayName": "Your gender",
      "type": "str",
      "listOptions": [
            {"label": "Male", "value": "male"},
            {"label": "Female", "value": "female"},
            {"label": "Other", "value": "other"}
        ],
```

```
      "isOptional": true
    },
    {
      "name": "country",
      "displayName": "Your origin country ?",
      "type": "list",
      "listOptions": "Afghanistan,Albania,Algeria,American
 Samoa,Andorra,Angola,Anguilla,Antarctica,Antigua and
 Barbuda,Argentina,Armenia,Aruba,Australia,Austria,Azerbaijan,Bahamas,Bahrain,Bangladesh,Barbad
 and Herzegovina,Botswana,Bouvet Island,Brazil,British Indian Ocean Territory,Brunei
 Darussalam,Bulgaria,Burkina Faso,Burundi,Cambodia,Cameroon,Canada,Cape
 Verde,Cayman Islands,Central African Republic,Chad,Chile,China,Christmas
 Island,Cocos (Keeling Islands),Colombia,Comoros,Congo,Cook Islands,Costa
 Rica,Cote D'Ivoire (Ivory Coast),Croatia (Hrvatska,Cuba,Cyprus,Czech
 Republic,Denmark,Djibouti,Dominica,Dominican Republic,East Timor,Ecuador,Egypt,El
 Salvador,Equatorial Guinea,Eritrea,Estonia,Ethiopia,Falkland Islands (Malvinas),Faroe
 Islands,Fiji,Finland,France,France,Metropolitan,French Guiana,French Polynesia,French
 Southern
 Territories,Gabon,Gambia,Georgia,Germany,Ghana,Gibraltar,Greece,Greenland,Grenada,Guadeloupe,G
Bissau,Guyana,Haiti,Heard and McDonald Islands,Honduras,Hong
 Kong,Hungary,Iceland,India,Indonesia,Iran,Iraq,Ireland,Israel,Italy,Jamaica,Japan,Jordan,Kazak
 (North),Korea
 (South),Kuwait,Kyrgyzstan,Laos,Latvia,Lebanon,Lesotho,Liberia,Libya,Liechtenstein,Lithuania,Lu
 Islands,Martinique,Mauritania,Mauritius,Mayotte,Mexico,Micronesia,Moldova,Monaco,Mongolia,Mont
 Antilles,New Caledonia,New Zealand,Nicaragua,Niger,Nigeria,Niue,Norfolk
 Island,Northern Mariana Islands,Norway,Oman,Pakistan,Palau,Panama,Papua
 New Guinea,Paraguay,Peru,Philippines,Pitcairn,Poland,Portugal,Puerto
 Rico,Qatar,Reunion,Romania,Russian Federation,Rwanda,Saint Kitts and Nevis,Saint
 Lucia,Saint Vincent and The Grenadines,Samoa,San Marino,Sao Tome and Principe,Saudi
 Arabia,Senegal,Seychelles,Sierra Leone,Singapore,Slovak Republic,Slovenia,Solomon
 Islands,Somalia,South Africa,S. Georgia and S. Sandwich Isls.,Spain,Sri
 Lanka,St. Helena,St. Pierre and Miquelon,Sudan,Suriname,Svalbard and Jan Mayen
 Islands,Swaziland,Sweden,Switzerland,Syria,Tajikistan,Tanzania,Thailand,Togo,Tokelau,Tonga,Tri
 and Tobago,Tunisia,Turkey,Turkmenistan,Turks and Caicos
 Islands,Tuvalu,Uganda,Ukraine,United Arab Emirates,United Kingdom
 (Britain / UK),United States of America (USA),US Minor Outlying
 Islands,Uruguay,Uzbekistan,Vanuatu,Vatican City State (Holy See),Venezuela,Viet
 Nam,Virgin Islands (British),Virgin Islands (US),Wallis and Futuna Islands,Western
 Sahara,Yemen,Yugoslavia,Zaire,Zambia,Zimbabwe",
      "description": "What country were you born in?",
      "listType": "str",
      "isOptional": true
    },
    {
```

```
      "name": "promotion",
      "displayName": "Do you want to receive promotional newsletter from us?",
      "type": "bool",
      "isOptional": true
    }
  ]
}
```

Ejemplos de scripts visuales personalizados

En los siguientes ejemplos, se llevan a cabo transformaciones equivalentes. Sin embargo, el segundo ejemplo (SparkSQL) es el más limpio y eficiente, seguido de la UDF de Pandas y, finalmente, las asignaciones de nivel bajo del primer ejemplo. El siguiente ejemplo es un ejemplo completo de una transformación sencilla para agregar dos columnas:

```
from awsglue import DynamicFrame

# You can have other auxiliary variables, functions or classes on this file, it won't
 affect the runtime
def record_sum(rec, col1, col2, resultCol):
    rec[resultCol] = rec[col1] + rec[col2]
    return rec


# The number and name of arguments must match the definition on json config file
# (expect self which is the current DynamicFrame to transform
# If an argument is optional, you need to define a default value here
#  (resultCol in this example is an optional argument)
def custom_add_columns(self, col1, col2, resultCol="result"):
    # The mapping will alter the columns order, which could be important
    fields = [field.name for field in self.schema()]
    if resultCol not in fields:
        # If it's a new column put it at the end
        fields.append(resultCol)
    return self.map(lambda record: record_sum(record, col1, col2,
 resultCol)).select_fields(paths=fields)


# The name we assign on DynamicFrame must match the configured "functionName"
DynamicFrame.custom_add_columns = custom_add_columns
```

El siguiente ejemplo es una transformación equivalente que aprovecha la API de SparkSQL.

```
from awsglue import DynamicFrame

# The number and name of arguments must match the definition on json config file
# (expect self which is the current DynamicFrame to transform
# If an argument is optional, you need to define a default value here
#  (resultCol in this example is an optional argument)
def custom_add_columns(self, col1, col2, resultCol="result"):
    df = self.toDF()
    return DynamicFrame.fromDF(
        df.withColumn(resultCol, df[col1] + df[col2]) # This is the conversion logic
        , self.glue_ctx, self.name)


# The name we assign on DynamicFrame must match the configured "functionName"
DynamicFrame.custom_add_columns = custom_add_columns
```

En el siguiente ejemplo, se usa las mismas transformaciones, pero con una UDF de Pandas, que es más eficiente que usar una UDF simple. Para obtener más información sobre cómo escribir UDF de Pandas, consulte la documentación de Apache Spark SQL.

```
from awsglue import DynamicFrame
import pandas as pd
from pyspark.sql.functions import pandas_udf

# The number and name of arguments must match the definition on json config file
# (expect self which is the current DynamicFrame to transform
# If an argument is optional, you need to define a default value here
#  (resultCol in this example is an optional argument)
def custom_add_columns(self, col1, col2, resultCol="result"):
    @pandas_udf("integer")  # We need to declare the type of the result column
    def add_columns(value1: pd.Series, value2: pd.Series) # pd.Series:
        return value1 + value2

    df = self.toDF()
    return DynamicFrame.fromDF(
        df.withColumn(resultCol, add_columns(col1, col2)) # This is the conversion
 logic
        , self.glue_ctx, self.name)

# The name we assign on DynamicFrame must match the configured "functionName"
```

```
DynamicFrame.custom_add_columns = custom_add_columns
```

Vídeo

En el siguiente video, se ofrece una introducción a las transformaciones visuales personalizadas y se muestra cómo usarlas.

Uso de marcos de lagos de datos con AWS Glue Studio

Información general

Los marcos de lago de datos de código abierto simplifican el procesamiento progresivo de datos para los archivos almacenados en lagos de datos creados en Amazon S3. La versión 3.0 y versiones posteriores de AWS Glue admiten los siguientes marcos de almacenamiento de lagos de datos de código abierto:

- Apache Hudi
- Linux Foundation Delta Lake
- Apache Iceberg

A partir de la versión de AWS Glue 4.0, AWS Glue proporciona compatibilidad nativa con estos marcos para que pueda leer y escribir los datos que almacene en Amazon S3 de una manera consistente entre transacciones. No es necesario instalar ningún conector independiente ni completar pasos de configuración adicionales para utilizar estos marcos en trabajos de AWS Glue.

Los marcos de lagos de datos se pueden utilizar como origen o destino en AWS Glue Studio a través de trabajos del editor de scripts de Spark. Para obtener más información sobre el uso de Apache Hudi, Apache Iceberg y Delta Lake, consulte Uso de marcos de lagos de datos con trabajos de ETL de AWS Glue.

Creación de formatos de tabla abiertos a partir de un origen de streaming de AWS Glue

Los trabajos ETL en streaming de AWS Glue consumen continuamente datos de los orígenes de streaming, los limpian y transforman durante el tránsito y los ponen a disposición para su análisis en cuestión de segundos.

AWS ofrece una amplia selección de servicios para satisfacer sus necesidades. Un servicio de replicación de bases de datos, como AWS Database Migration Service, puede replicar los datos de sus sistemas de origen a Amazon S3, que normalmente aloja la capa de almacenamiento del lago de datos. Si bien es sencillo aplicar las actualizaciones en un sistema de administración de base de datos relacional (RDBMS) que respalda una aplicación de origen en línea, es difícil aplicar este proceso de CDC en sus lagos de datos. Los marcos de administración de datos de código abierto simplifican el procesamiento progresivo de datos y el desarrollo de canalizaciones de datos, y son una buena opción para resolver este problema.

Para obtener más información, consulte:

- Cree un lago de datos transaccional casi en tiempo real basado en Apache Hudi mediante AWS Glue Streaming
- Crear un lago de datos de Apache Iceberg en tiempo real alineado con el RGPD

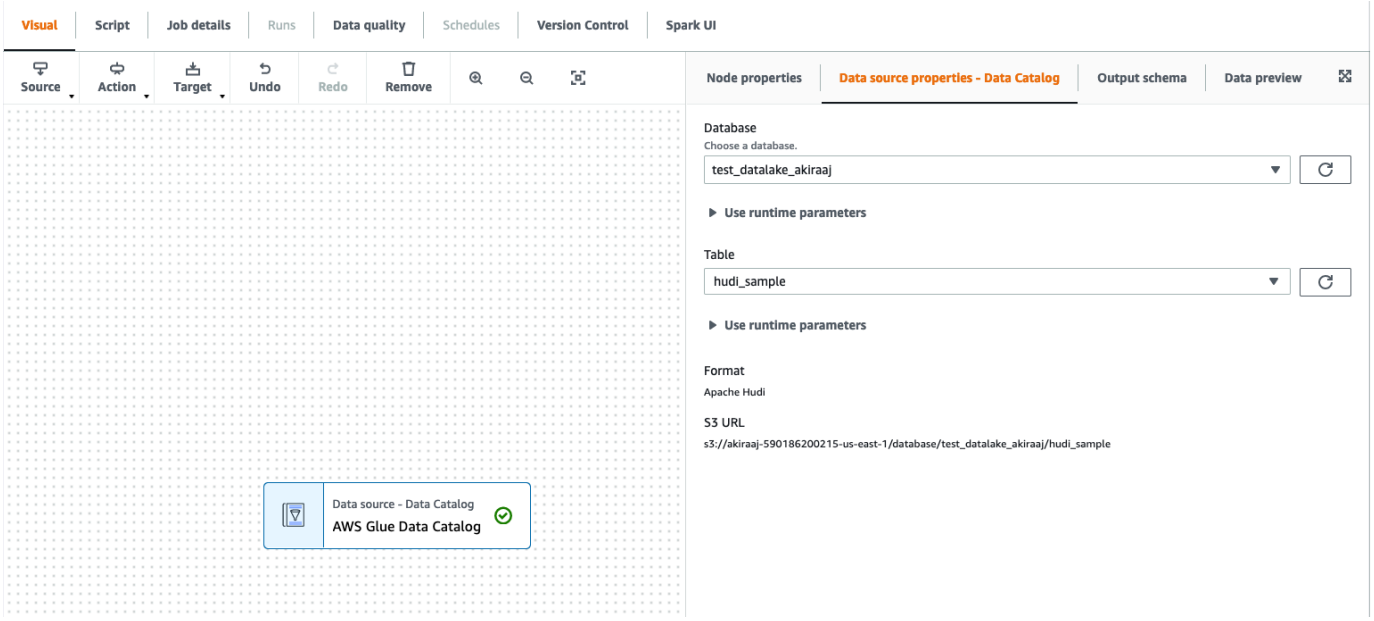
Uso del marco de Hudi en AWS Glue Studio

Cuando crea o edita un trabajo, AWS Glue Studio agrega automáticamente las bibliotecas de Hudi correspondientes en su nombre según la versión de AWS Glue que utilice. Para obtener más información, consulte Uso del marco de Hudi en AWS Glue.

Uso del marco de Apache Hudi en los orígenes de datos del Catálogo de datos

Para agregar un formato de origen de datos de Hudi a un trabajo, haga lo siguiente:

1. En el menú Origen, seleccione Catálogo de datos de AWS Glue Studio.
2. En la pestaña Propiedades del origen de datos, elija una base de datos y una tabla.
3. AWS Glue Studio muestra el tipo de formato como Apache Hudi y la dirección URL de Amazon S3.



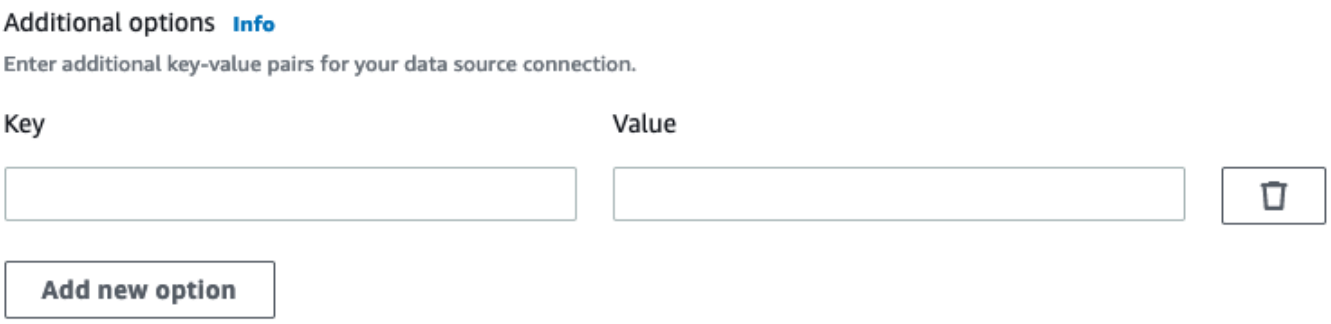
Uso del marco de Hudi en orígenes de datos de Amazon S3

1. En el menú Origen, elija Amazon S3.
2. Si elige la tabla del Catálogo de datos como el tipo de origen de Amazon S3, elija una base de datos y una tabla.
3. AWS Glue Studio muestra el formato como Apache Hudi y la dirección URL de Amazon S3.
4. Si elige la ubicación de Amazon S3 como el Tipo de origen de Amazon S3, elija la dirección URL de Amazon S3 haciendo clic en Examinar Amazon S3.
5. En Formato de datos, seleccione Apache Hudi.

> **Note**
>
> Si AWS Glue Studio no puede deducir el esquema de la carpeta o el archivo de Amazon S3 que ha seleccionado, elija Opciones adicionales para seleccionar una nueva carpeta o archivo.
> En Opciones adicionales, elija una de las siguientes opciones en Inferencia del esquema:
>
> - Permitir que AWS Glue Studio elija automáticamente un archivo de muestra: AWS Glue Studio elegirá un archivo de muestra en la ubicación de Amazon S3 para poder deducir el esquema. En el campo Archivo de muestra seleccionado de manera automática, puede ver el archivo que se seleccionó automáticamente.

- Elegir un archivo de muestra de Amazon S3: elija el archivo de Amazon S3 que va a utilizar haciendo clic en Examinar Amazon S3.

6. Haga clic en Deducir esquema. A continuación, puede ver el esquema de salida haciendo clic en la pestaña Esquema de salida.
7. Elija Opciones adicionales para introducir un par clave-valor.



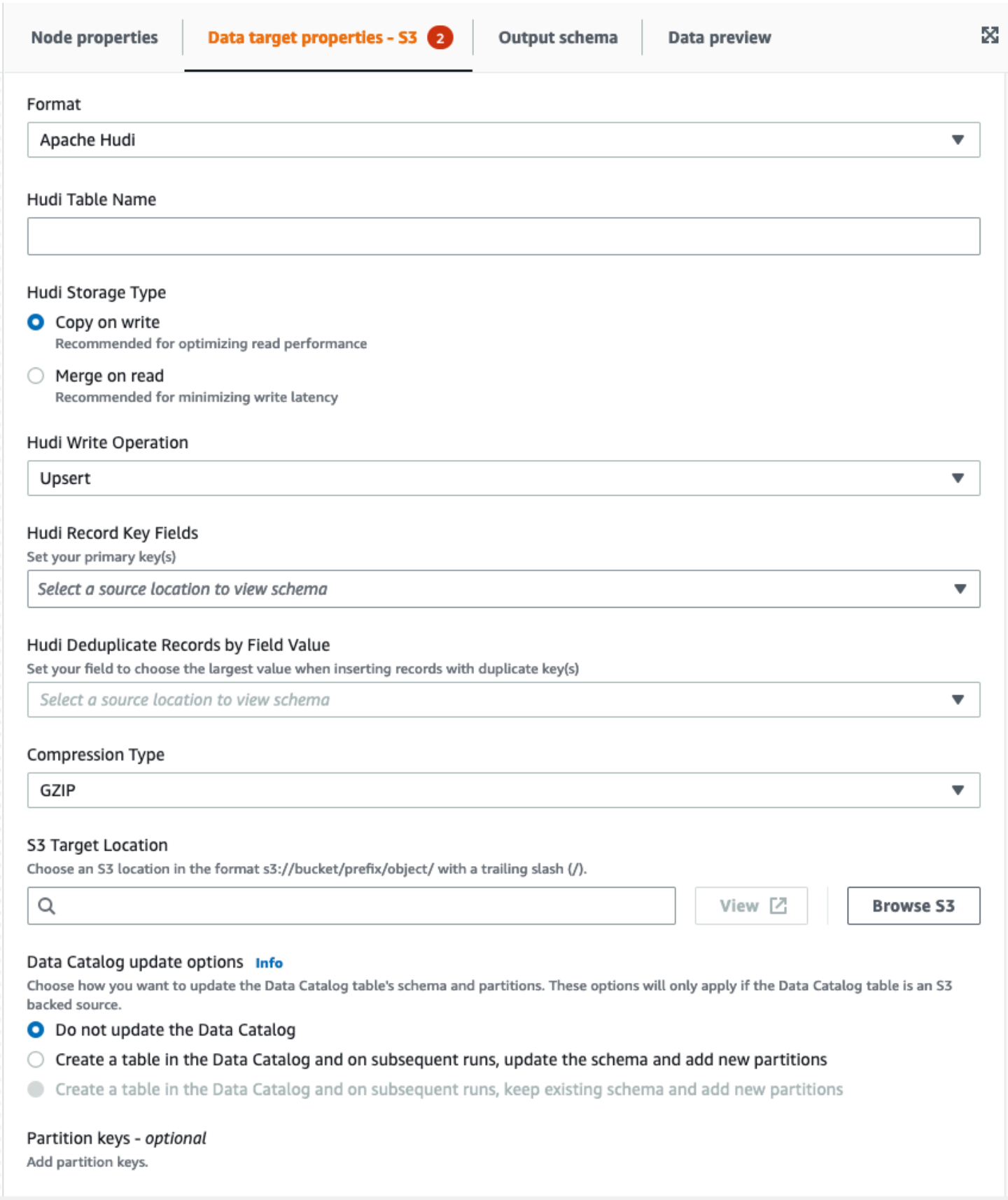
Uso del marco de Apache Hudi en los destinos de datos

Uso del marco de Apache Hudi en los destinos de datos del Catálogo de datos

1. En el menú Destino, seleccione Catálogo de datos de AWS Glue Studio.
2. En la pestaña Propiedades del origen de datos, elija una base de datos y una tabla.
3. AWS Glue Studio muestra el tipo de formato como Apache Hudi y la dirección URL de Amazon S3.

Uso del marco de Apache Hudi en los destinos de datos de Amazon S3

Introduzca valores o seleccione una de las opciones disponibles para configurar el formato Apache Hudi. Para obtener más información sobre Apache Hudi, consulte la Documentación de Apache Hudi.


Node properties
Data target properties - S3 2
Output schema
Data preview
Format
Apache Hudi
Hudi Table Name
Hudi Storage Type
Copy on write
Recommended for optimizing read performance
Merge on read
Recommended for minimizing write latency
Hudi Write Operation
Upsert
Hudi Record Key Fields
Set your primary key(s)
Select a source location to view schema
Hudi Deduplicate Records by Field Value
Set your field to choose the largest value when inserting records with duplicate key(s)
Select a source location to view schema
Compression Type
GZIP
S3 Target Location
Choose an S3 location in the format s3://bucket/prefix/object/ with a trailing slash (/).
View
Browse S3
Data Catalog update options Info
Choose how you want to update the Data Catalog table's schema and partitions. These options will only apply if the Data Catalog table is an S3 backed source.
Do not update the Data Catalog
Create a table in the Data Catalog and on subsequent runs, update the schema and add new partitions
Create a table in the Data Catalog and on subsequent runs, keep existing schema and add new partitions
Partition keys - optional
Add partition keys.

- Nombre de la tabla de Hudi: este es el nombre de su tabla de Hudi.
- Tipo de almacenamiento de Hudi: elija entre dos opciones.
  - Copiar durante la escritura: se recomienda esta opción para optimizar el rendimiento de lectura. Este es el tipo de almacenamiento predeterminado de Hudi. Cada actualización crea una versión nueva de los archivos durante la escritura.
  - Combinar durante la lectura: se recomienda esta opción para minimizar la latencia de escritura. Las actualizaciones se registran en archivos delta basados en filas y se compactan según sea necesario para crear nuevas versiones de los archivos en columnas.
- Operación de escritura de Hudi: elija entre las siguientes opciones.
  - Upsert: esta es la operación predeterminada en la que los registros de entrada se etiquetan primero como inserciones o actualizaciones al consultar el índice. Se recomienda para la actualización de los datos existentes.
  - Insert: esta operación inserta registros, pero no comprueba si hay registros existentes, lo que puede generar duplicados.
  - Bulk Insert: esta operación inserta registros y se recomienda para grandes cantidades de datos.
- Campos de clave de registro de Hudi: utilice la barra de búsqueda para ubicar y elegir las claves de registro principales. Los registros en Hudi se identifican mediante una clave principal, que representa la combinación de una clave de registro y una ruta de partición a la que pertenece el registro.
- Campo de combinación previa de Hudi: este es el campo que se utiliza en la combinación previa antes de escribir realmente. Cuando dos registros tienen el mismo valor para la clave, AWS Glue Studio selecciona el que tiene el valor más alto para el campo de combinación previa. Establezca un campo con un valor incremental (por ejemplo, updated_at).
- Tipo de compresión: elija una de las opciones de tipo de compresión; sin comprimir, GZIP, LZO o Snappy.
- Ubicación de destino de Amazon S3: para elegir la ubicación de destino de Amazon S3, haga clic en Examinar S3.
- Opciones de actualización del Catálogo de datos: elija entre siguientes opciones.
  - Do not update the Data Catalog (No actualizar el Catálogo de datos): (predeterminado) seleccione esta opción si no desea que el trabajo actualice el Catálogo de datos, incluso si el esquema cambia o se agregan nuevas particiones.
  - Crear una tabla en el Catálogo de datos y en las ejecuciones posteriores, actualizar el esquema y agregar nuevas particiones: si elige esta opción, el trabajo crea la tabla en el Catálogo de datos durante la primera ejecución del trabajo. En las ejecuciones de trabajos posteriores, el

trabajo actualiza la tabla del Catálogo de datos si cambia el esquema o se agregan nuevas particiones.

También debe seleccionar una base de datos en el Catálogo de datos e introducir un nombre de tabla.

- Create a table in the Data Catalog and on subsequent runs, keep existing schema and add new partitions (Crear una tabla en el Catálogo de datos y en ejecuciones posteriores, mantener el esquema existente y agregar nuevas particiones): si elige esta opción, el trabajo crea la tabla en el Catálogo de datos durante la primera ejecución del trabajo. En las ejecuciones de trabajos posteriores, el trabajo actualiza la tabla del Catálogo de datos solo para agregar nuevas particiones.

  También debe seleccionar una base de datos en el Catálogo de datos e introducir un nombre de tabla.

- Partition keys (Claves de partición): elija qué columnas utilizar como claves de partición en la salida. Para agregar más claves de partición, elija Add a partition key (Agregar una clave de partición).
- Opciones adicionales: introduzca un par clave-valor según sea necesario.

Generación de código a través de AWS Glue Studio

Cuando se guarda el trabajo, los siguientes parámetros se agregan al trabajo si se detecta un origen o un destino de Hudi:

- `--datalake-formats`: una lista independiente de los formatos de lagos de datos detectados en el trabajo visual (ya sea eligiendo un “Formato” directamente o seleccionando una tabla del catálogo que cuenta con el respaldo de un lago de datos indirectamente).
- `--conf` : se genera en función del valor de `--datalake-formats`. Por ejemplo, si el valor de `--datalake-formats` es “hudi”, AWS Glue genera un valor de `spark.serializer=org.apache.spark.serializer.KryoSerializer –conf spark.sql.hive.convertMetastoreParquet=false` para este parámetro.

Anulación de bibliotecas proporcionadas por AWS Glue

Para usar una versión de Hudi no compatible con AWS Glue, puede especificar sus propios archivos JAR de la biblioteca de Hudi. Para usar su propio archivo JAR, haga lo siguiente:

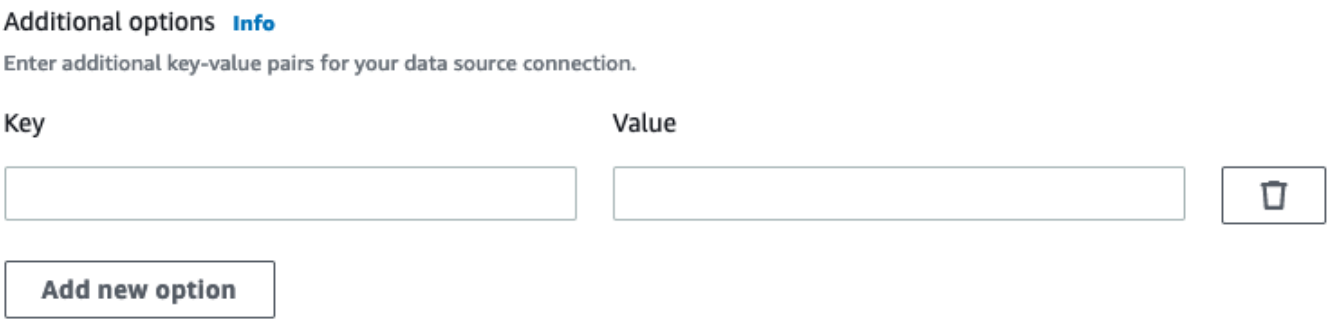
- Utilice el parámetro de trabajo `--extra-jars`. Por ejemplo, `'--extra-jars': 's3pathtojarfile.jar'`. Para obtener más información, consulte Parámetros de los trabajos de AWS Glue.
- No incluya `hudi` como valor para el parámetro de trabajo `--datalake-formats`. Al introducir una cadena en blanco como valor, se asegura de que AWS Glue no le proporcione automáticamente ninguna biblioteca de lagos de datos. Para obtener más información, consulte Uso del marco de Hudi en AWS Glue.

Uso del marco de Delta Lake en AWS Glue Studio

Uso del marco de Delta Lake en los orígenes de datos

Uso del marco de Delta Lake en los orígenes de datos de Amazon S3

1. En el menú Origen, elija Amazon S3.
2. Si elige la tabla del Catálogo de datos como el tipo de origen de Amazon S3, elija una base de datos y una tabla.
3. AWS Glue Studio muestra el formato como Delta Lake y la dirección URL de Amazon S3.
4. Elija Opciones adicionales para introducir un par clave-valor. Por ejemplo, un par clave-valor podría ser el siguiente: clave: timestampAsOf y valor: 2023-02-24 14:16:18.



5. Si elige la ubicación de Amazon S3 como el Tipo de origen de Amazon S3, elija la dirección URL de Amazon S3 haciendo clic en Examinar Amazon S3.
6. En Formato de datos, elija Delta Lake.

> **Note**
>
> Si AWS Glue Studio no puede deducir el esquema de la carpeta o el archivo de Amazon S3 que ha seleccionado, elija Opciones adicionales para seleccionar una nueva carpeta o archivo.
> En Opciones adicionales, elija una de las siguientes opciones en Inferencia del esquema:
>
> - Permitir que AWS Glue Studio elija automáticamente un archivo de muestra: AWS Glue Studio elegirá un archivo de muestra en la ubicación de Amazon S3 para poder deducir el esquema. En el campo Archivo de muestra seleccionado de manera automática, puede ver el archivo que se seleccionó automáticamente.
> - Elegir un archivo de muestra de Amazon S3: elija el archivo de Amazon S3 que va a utilizar haciendo clic en Examinar Amazon S3.

7. Haga clic en Deducir esquema. A continuación, puede ver el esquema de salida haciendo clic en la pestaña Esquema de salida.

Uso del marco de Delta Lake en los orígenes de datos del Catálogo de datos

1. En el menú Origen, seleccione Catálogo de datos de AWS Glue Studio.
2. En la pestaña Propiedades del origen de datos, elija una base de datos y una tabla.
3. AWS Glue Studio muestra el tipo de formato como Delta Lake y la dirección URL de Amazon S3.

> **Note**
>
> Si su origen de Delta Lake aún no está registrado como tabla del Catálogo de datos AWS Glue, tiene dos opciones:
>
> 1. Cree un rastreador de AWS Glue para el almacén de datos de Delta Lake. Para obtener más información, consulte Cómo especificar opciones de configuración para un almacén de datos de Delta Lake.
> 2. Use un origen de datos de Amazon S3 para seleccionar su origen de datos de Delta Lake. Consulte Uso del marco de Delta Lake en los orígenes de datos de Amazon S3 .

Uso de formatos de Delta Lake en los destinos de datos

Uso de formatos de Delta Lake en los destinos de datos del Catálogo de datos

1. En el menú Destino, seleccione Catálogo de datos de AWS Glue Studio.
2. En la pestaña Propiedades del origen de datos, elija una base de datos y una tabla.
3. AWS Glue Studio muestra el tipo de formato como Delta Lake y la dirección URL de Amazon S3.

Uso de formatos de Delta Lake en los orígenes de datos de Amazon S3

Introduzca valores o seleccione una de las opciones disponibles para configurar el formato de Delta Lake.

- Tipo de compresión: elija una de las opciones de tipo de compresión; sin comprimir o Snappy.
- Ubicación de destino de Amazon S3: para elegir la ubicación de destino de Amazon S3, haga clic en Examinar S3.
- Opciones de actualización del Catálogo de datos: en el editor visual de Glue Studio, no se admite la actualización del Catálogo de datos para este formato.
  - Do not update the Data Catalog (No actualizar el Catálogo de datos): (predeterminado) seleccione esta opción si no desea que el trabajo actualice el Catálogo de datos, incluso si el esquema cambia o se agregan nuevas particiones.
  - Para actualizar el Catálogo de datos después de la ejecución del trabajo de AWS Glue, ejecute o programe un rastreador de AWS Glue. Para obtener más información, consulte Cómo especificar opciones de configuración para un almacén de datos de Delta Lake.
- Claves de partición: elija qué columnas utilizar como claves de partición en la salida. Para agregar más claves de partición, elija Agregar una clave de partición.
- Puede elegir Opciones adicionales para introducir un par clave-valor. Por ejemplo, un par clave-valor podría ser el siguiente: clave: timestampAsOf y valor: 2023-02-24 14:16:18.

Uso del marco Apache Iceberg en AWS Glue Studio

Uso del marco de Apache Iceberg en los destinos de datos

Uso del marco de Apache Iceberg en los destinos de datos del Catálogo de datos

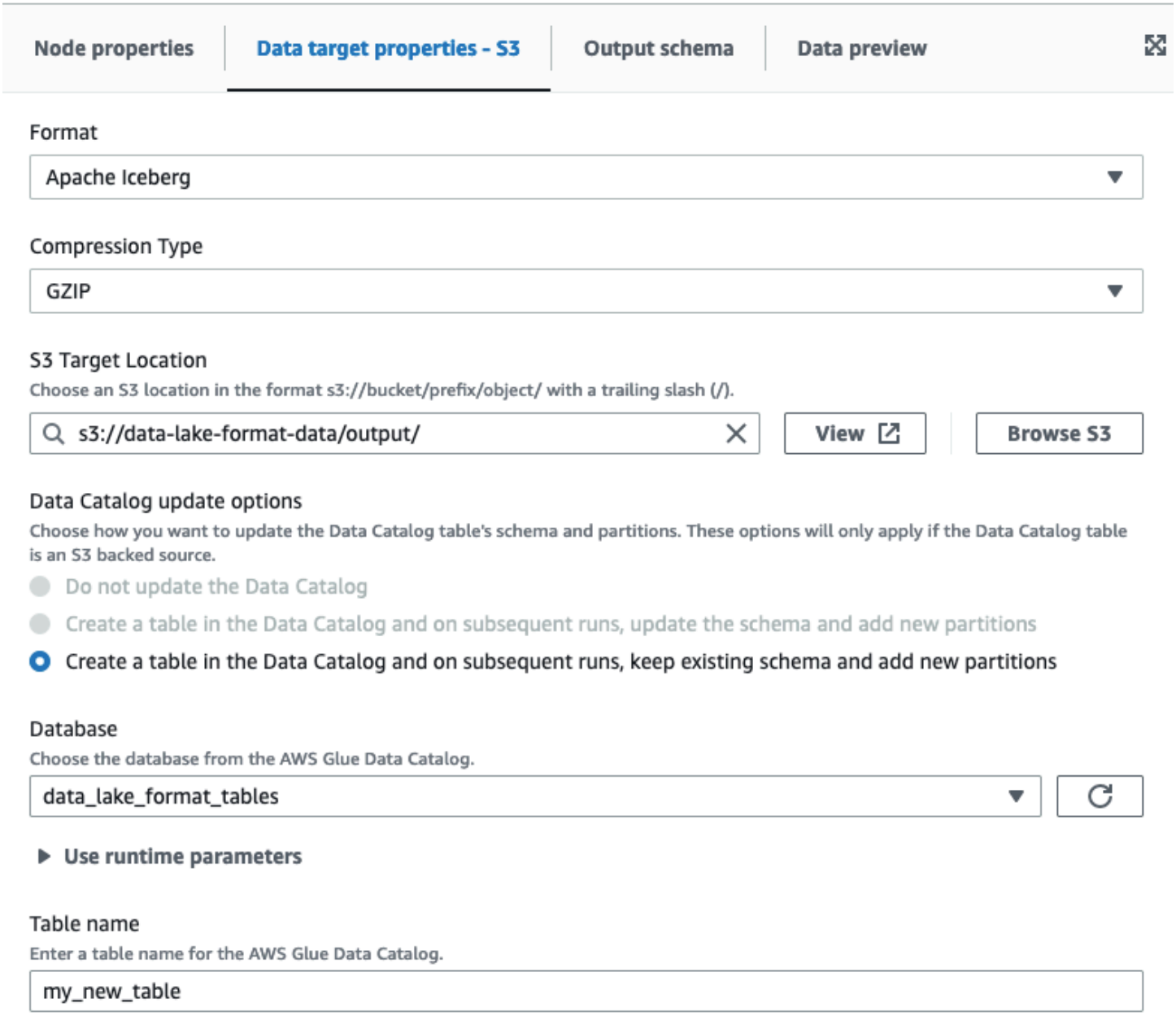
1. En el menú Destino, seleccione Catálogo de datos de AWS Glue Studio.

2. En la pestaña Propiedades del origen de datos, elija una base de datos y una tabla.
3. AWS Glue Studio muestra el tipo de formato como Apache Iceberg y la dirección URL de Amazon S3.

Uso del marco de Apache Iceberg en los destinos de datos de Amazon S3

Ingrese valores o seleccione una de las opciones disponibles para configurar el formato Apache Iceberg.

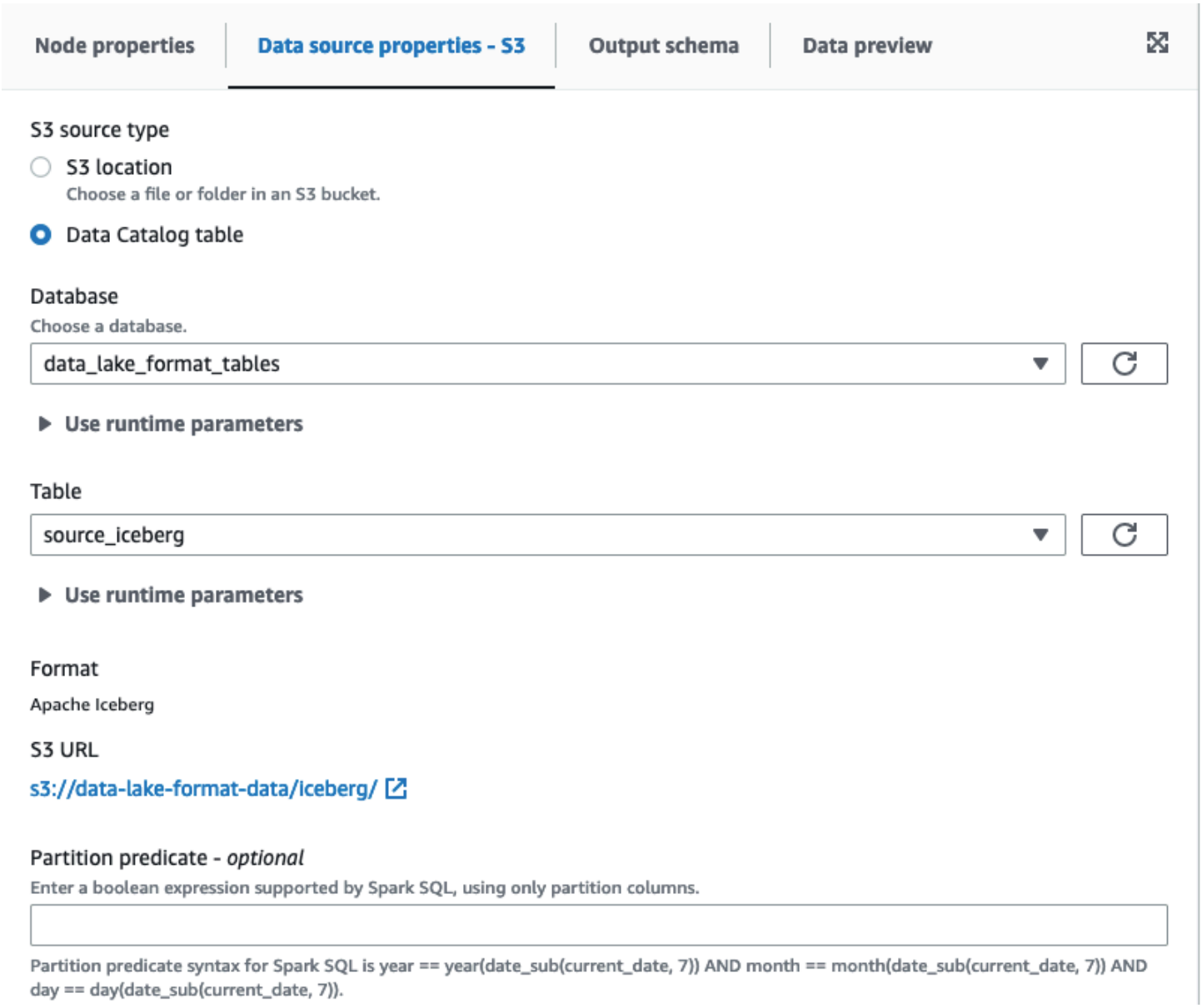
- Formato: elija Apache Iceberg en el menú desplegable.
- Ubicación de destino de Amazon S3: para elegir la ubicación de destino de Amazon S3, haga clic en Examinar S3.
- Opciones de actualización del Catálogo de datos: cree una tabla en el Catálogo de datos y en las ejecuciones posteriores, mantenga el esquema existente y agregue nuevas particiones. Debe estar seleccionado para continuar. Para escribir una nueva tabla Iceberg con AWS Glue requiere que Data Catalog esté configurado como catálogo para la tabla de Iceberg. Para actualizar una tabla de Iceberg existente que se haya registrado en Data Catalog, elija Data Catalog como destino.
  - Base de datos: elija la base de datos de Data Catalog.
  - Table name: escriba un nombre único para la tabla. Los nombres de las tablas de Apache Iceberg deben estar en minúsculas. Utilice guiones bajos si es necesario, ya que no se permiten espacios. Por ejemplo, “data_lake_format_tables”.



Uso del marco de Apache Iceberg en los orígenes de datos de Amazon S3

Uso del marco de Apache Iceberg en los orígenes de datos del Catálogo de datos

1. En el menú Origen, seleccione Catálogo de datos de AWS Glue Studio.
2. En la pestaña Propiedades del origen de datos, elija una base de datos y una tabla.
3. AWS Glue Studio muestra el tipo de formato como Apache Iceberg y la dirección URL de Amazon S3.



Uso del marco de Apache Iceberg en los orígenes de datos de Amazon S3

Apache Iceberg no está disponible como opción de datos para los nodos de origen de Amazon S3 en AWS Glue Studio.

Configuración de nodos de destino de datos

El destino de datos es donde el trabajo escribe los datos transformados.

Información general de las opciones de destino de datos

Su destino de datos (también denominado receptor de datos) puede ser uno de los siguientes:

- S3: el trabajo escribe los datos en un archivo en la ubicación de Amazon S3 que elija y en el formato que especifique.

  Si configura columnas de partición para el destino de datos, el trabajo escribe el conjunto de datos en Amazon S3 en directorios en función de la clave de partición.
- AWS Glue Data Catalog: el trabajo utiliza la información asociada a la tabla del Catálogo de datos para escribir los datos de salida en una ubicación de destino.

  Puede crear la tabla en forma manual o con el rastreador. También puede utilizar plantillas de AWS CloudFormation para crear tablas en el Catálogo de datos.
- Conector: un conector es un fragmento de código que facilita la comunicación entre el almacén de datos y AWS Glue. El trabajo utiliza el conector y la conexión asociada para escribir los datos de salida en una ubicación de destino. Puede suscribirse a un conector ofrecido en AWS Marketplace o puede crear su propio conector personalizado. Para obtener más información, consulte Agregar conectores a AWS Glue Studio.

Puede elegir actualizar el Catálogo de datos cuando su trabajo escriba en un destino de datos de Amazon S3. En lugar de requerir que un rastreador actualice el Catálogo de datos cuando cambian el esquema o las particiones, esta opción facilita la actualización de las tablas. Esta opción simplifica el proceso de hacer que los datos estén disponibles para análisis mediante el agregado opcional de nuevas tablas al Catálogo de datos, la actualización de particiones de tabla y del esquema de las tablas directamente desde el trabajo.

Edición del nodo de destino de datos

El destino de datos es donde el trabajo escribe los datos transformados.

Para agregar o configurar un nodo de destino de datos en el diagrama de trabajo

1. (Opcional) si necesita agregar un nodo de destino, elija Target (Destino) en la barra de herramientas situada en la parte superior del editor visual y elija S3 o Glue Data Catalog (Catálogo de datos de Glue).

   - Si elige S3 para el destino, el trabajo escribe el conjunto de datos en uno o más archivos en la ubicación de Amazon S3 que especifique.
   - Si elige AWS Glue Data Catalog para el destino, el trabajo escribe en una ubicación descrita por la tabla seleccionada en el Catálogo de datos.

2. Elija un nodo de destino de datos en el diagrama de trabajo. Al elegir un nodo, aparece el panel de detalles del nodo en el lado derecho de la página.
3. Elija la pestaña Node properties (Propiedades del nodo) y, a continuación, escriba la información siguiente:

   - Name (Nombre): ingrese un nombre para asociar al nodo en el diagrama de trabajo.
   - Node type (Tipo de nodo): ya se debe haber seleccionado un valor, pero podrá modificarlo según sea necesario.
   - Node parents (Nodo principales): el nodo principal es el nodo del diagrama de trabajo que proporciona los datos de salida que desea escribir en la ubicación de destino. Para un diagrama de trabajo previamente completado, el nodo de destino ya debe tener seleccionado el nodo principal. Si no se muestra ningún nodo principal, elija un nodo principal de la lista.

     Un nodo de destino tiene un nodo principal único.
4. Configure la información de Data source properties (Propiedades de origen de datos). Para obtener más información, consulte las siguientes secciones:

   - Uso de Amazon S3 para el destino de datos
   - Uso de tablas del Catálogo de datos para el destino de datos
   - Uso de un conector para el destino de datos
5. (Opcional) después de configurar las propiedades del nodo de destino de datos, puede ver el esquema de salida para sus datos al seleccionar la pestaña Output schema (Esquema de salida) en el panel de detalles del nodo. La primera vez que elija esta pestaña para cualquier nodo de trabajo, se le pedirá que proporcione un rol de IAM para acceder a los datos. Si no ha especificado un rol de IAM en Job details (Detalles del trabajo), se le pedirá que ingrese un rol de IAM aquí.

Uso de Amazon S3 para el destino de datos

En todos los orígenes de datos, excepto Amazon S3 y los conectores, debe existir una tabla en el AWS Glue Data Catalog para el tipo de fuente que elija. AWS Glue Studio no crea la tabla del Catálogo de datos.

Para configurar un nodo de destino de datos que escriba en Amazon S3

1. Vaya al editor visual para acceder a un trabajo nuevo o guardado.
2. Elija un nodo de origen de datos en el diagrama de trabajo.

3. Elija la pestaña Data source properties (Propiedades de origen de datos) y, a continuación, escriba la información siguiente:

   - Format (Formato): elija un formato de la lista. Los tipos de formato disponibles para los resultados de datos son:
     - JSON: notación de objetos JavaScript.
     - CSV: valores separados por comas.
     - Avro: Apache Avro JSON binario.
     - Parquet: almacenamiento en columna de Apache Parquet.
     - Glue Parquet: un tipo personalizado de escritor de Parquet, optimizado para `DynamicFrames` como el formato de datos. En lugar de requerir un esquema precalculado para los datos, calcula y modifica el esquema dinámicamente.
     - ORC: formato Apache Optimized Row Columnar (ORC).

     Para obtener más información sobre estas opciones de formato, consulte Opciones de formato para las entradas y salidas de ETL en AWS Glue en la Guía para desarrolladores de AWS Glue.
   - Compression Type (Tipo de compresión): puede optar por comprimir los datos de manera opcional mediante el formato `gzip` o `bzip2`. El valor predeterminado es no compresión, o None (Ninguno).
   - S3 Target Location (Ubicación de destino de S3): el bucket y la ubicación de Amazon S3 para la salida de datos. Puede elegir la opción Browse S3 (Examinar S3) para ver los depósitos de Amazon S3 a los que tiene acceso y elegir uno como destino.
   - Opciones de actualización del Catálogo de datos
     - Do not update the Data Catalog (No actualizar el Catálogo de datos): (predeterminado) seleccione esta opción si no desea que el trabajo actualice el Catálogo de datos, incluso si el esquema cambia o se agregan nuevas particiones.
     - Create a table in the Data Catalog and on subsequent runs, update the schema and add new partitions (Crear una tabla en el Catálogo de datos y en ejecuciones posteriores, actualizar el esquema y agregar nuevas particiones): si elige esta opción, el trabajo crea la tabla en el Catálogo de datos durante la primera ejecución del trabajo. En las ejecuciones de trabajos posteriores, el trabajo actualiza la tabla del Catálogo de datos si cambia el esquema o se agregan nuevas particiones.

También debe seleccionar una base de datos en el Catálogo de datos e introducir un nombre de tabla.

- Create a table in the Data Catalog and on subsequent runs, keep existing schema and add new partitions (Crear una tabla en el Catálogo de datos y en ejecuciones posteriores, mantener el esquema existente y agregar nuevas particiones): si elige esta opción, el trabajo crea la tabla en el Catálogo de datos durante la primera ejecución del trabajo. En las ejecuciones de trabajos posteriores, el trabajo actualiza la tabla del Catálogo de datos solo para agregar nuevas particiones.

  También debe seleccionar una base de datos en el Catálogo de datos e introducir un nombre de tabla.
- Partition keys (Claves de partición): elija qué columnas utilizar como claves de partición en la salida. Para agregar más claves de partición, elija Add a partition key (Agregar una clave de partición).

Uso de tablas del Catálogo de datos para el destino de datos

En todos los orígenes de datos, excepto Amazon S3 y los conectores, debe existir una tabla en el AWS Glue Data Catalog para el tipo de destino que elija. AWS Glue Studio no crea la tabla del Catálogo de datos.

Para configurar las propiedades de datos de un destino que utiliza una tabla del Catálogo de datos

1. Vaya al editor visual para acceder a un trabajo nuevo o guardado.
2. Elija un nodo de destino de datos en el diagrama de trabajo.
3. Elija la pestaña Data target properties (Propiedades de destino de datos) y, a continuación, escriba la información siguiente:

   - Database (Base de datos): elija la base de datos que contiene la tabla que desea utilizar como destino a partir de la lista. Esta base de datos ya debe existir en el Catálogo de datos.
   - Table (Tabla): elija la tabla que define el esquema de los datos de salida a partir de la lista. Esta tabla ya debe existir en el Catálogo de datos.

     Una tabla del Catálogo de datos está formada por los nombres de las columnas, las definiciones de tipos de datos, la información de partición y otros metadatos acerca de un conjunto de datos de destino. Su trabajo escribe a una ubicación descrita en esta tabla en el Catálogo de datos.

Para obtener más información acerca de cómo crear tablas en el Catálogo de datos, consulte Definición de tablas en el Catálogo de datos en la Guía para desarrolladores de AWS Glue.

- Opciones de actualización del Catálogo de datos
  - Do not change table definition (No cambiar la definición de la tabla): (predeterminado) seleccione esta opción si no desea que el trabajo actualice el Catálogo de datos, incluso si el esquema cambia o se agregan nuevas particiones.
  - Update schema and add new partitions (Actualizar esquema y agregar nuevas particiones): si elige esta opción, el trabajo actualiza la tabla del Catálogo de datos si cambia el esquema o se agregan nuevas particiones.
  - Keep existing schema and add new partitions (Mantener el esquema existente y agregar nuevas particiones): si elige esta opción, el trabajo actualiza la tabla del Catálogo de datos solo para agregar nuevas particiones.
  - Partition keys (Claves de partición): elija qué columnas utilizar como claves de partición en la salida. Para agregar más claves de partición, elija Add a partition key (Agregar una clave de partición).

Uso de un conector para el destino de datos

Si selecciona un conector para el Node type (Tipo de nodo), siga las instrucciones en Creación de trabajos con conectores personalizados para finalizar la configuración de las propiedades del destino de datos.

Edición o carga de un script de trabajo

Utilice el editor visual de AWS Glue Studio para editar el script de trabajo o cargar el propio.

Puede utilizar el editor visual para editar nodos de trabajo sólo si estos se crearon con AWS Glue Studio. Si el trabajo se creó con la consola de AWS Glue, a través de comandos de la API, o con la interfaz de línea de comandos (CLI), puede utilizar el editor de scripts en AWS Glue Studio para editar el script, los parámetros y la programación del trabajo. También puede editar el script para un trabajo creado en AWS Glue Studio mediante la conversión del trabajo a modo de sólo script.

Para editar el script de trabajo o cargar su propio script

1. Si crea un nuevo trabajo, en la página Jobs (Trabajos), seleccione la opción Spark script editor (Editor de scripts de Spark) para crear un trabajo de Spark o elija la opción Python Shell script editor (Editor de scripts de shell de Python) para crear un trabajo de shell de Python. Puede

escribir un nuevo script o cargar un script existente. Si elige Spark script editor (Editor de scripts de Spark), puede escribir o cargar un script de Scala o Python. Si elige Python Shell script editor (Editor de scripts de shell de Python), solo puede escribir o cargar un script de Python.

Después de elegir la opción para crear un nuevo trabajo, en la sección Options (Opciones) que aparece, puede optar por comenzar con un script de inicio (Create a new script with boilerplate code (Crear un nuevo script con código reutilizable), o puede cargar un archivo local para utilizarlo como script de trabajo.

Si elige Spark script editor (Editor de scripts de Spark), puede cargar archivos de script de Scala o Python. Los scripts de Scala deben tener la extensión de archivo `.scala`. Los scripts de Python deben reconocerse como archivos de tipo Python. Si elige Python Shell script editor (Editor de scripts de shell de Python), solo puede cargar archivos de script de Python.

Cuando termine de tomar sus decisiones, elija Create (Crear) para crear el trabajo y abrir el editor visual.

2. Diríjase al editor visual de trabajos para el trabajo nuevo o guardado y, a continuación, elija la pestaña Script.
3. Si no creó un nuevo trabajo con una de las opciones del editor de script y nunca ha editado el script para un trabajo existente, la pestaña Script muestra el encabezado Script (Locked) [Script (bloqueado)]. Esto significa que el editor de scripts está en modo de solo lectura. Seleccione Edit script (Editar script) para desbloquear el script para su edición.

   Para poder editar el script, AWS Glue Studio convierte el trabajo de visual a de sólo script. Si desbloquea el script para editarlo, no podrá utilizar el editor visual para este trabajo después de guardarlo.

   En la ventana de confirmación, elija Confirm (Confirmar) para continuar o Cancel (Cancelar) para mantener el trabajo disponible para la edición visual.

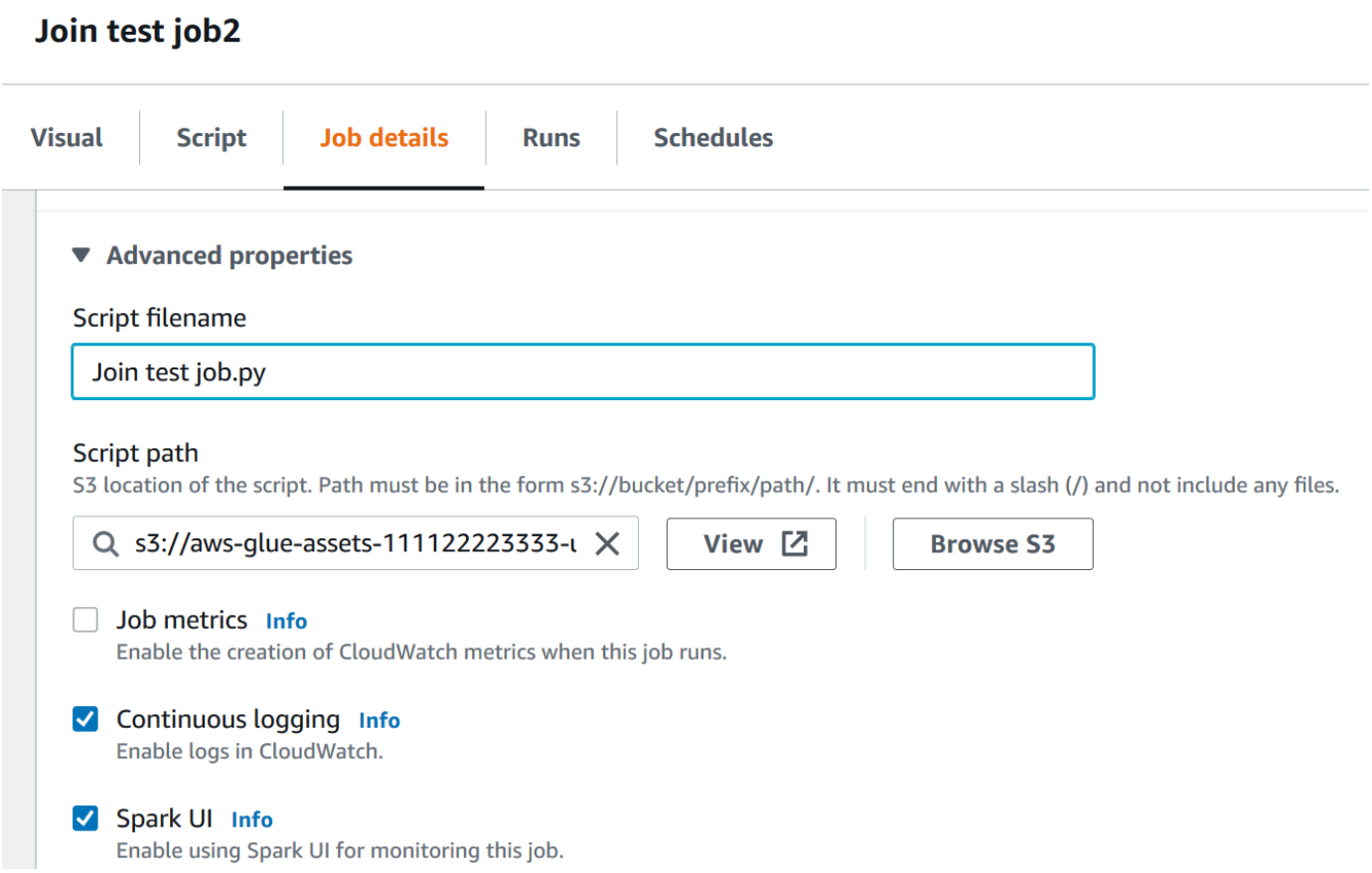
   Si elige Confirm (Confirmar), la pestaña Visual ya no aparecerá en el editor. Puede utilizar AWS Glue Studio para modificar el script mediante el editor de script, modificar los detalles o la programación del trabajo o ver las ejecuciones de trabajos.

> **Note**
>
> La conversión a un trabajo de sólo script solo será permanente una vez que guarde el trabajo. Si actualiza la página web de la consola o cierra el trabajo antes de guardarlo y vuelve a abrirlo en el editor visual, podrá editar los nodos individuales en el editor visual.

4. Edite el script según sea necesario.

   Cuando haya terminado de editar el script, elija Save (Guardar) para guardar el trabajo y convertir el trabajo de visual a sólo script de forma permanente.

5. (Opcional) Puede descargar el script desde la consola de AWS Glue Studio mediante el botón Download (Descargar) en la pestaña Script. Al seleccionar este botón, se abre una nueva ventana del navegador, que muestra el script desde su ubicación en Amazon S3. Los parámetros Script filename (Nombre de archivo del script) y Script path (Ruta del script) en la pestaña Job details (Detalles del trabajo) determinan el nombre y la ubicación del archivo de script en Amazon S3.



Cuando guarda el trabajo, AWS Glue guarda el script de trabajo en la ubicación especificada por estos campos. Si modifica el archivo de script en esta ubicación dentro de Amazon S3, AWS Glue Studio cargará el script modificado la próxima vez que edite el trabajo.

Creación y edición de scripts de Scala en AWS Glue Studio

Cuando elige el editor de script para crear un trabajo, de forma predeterminada, el lenguaje de programación de trabajo se establece en `Python 3`. Si elige escribir un nuevo script en lugar de cargarlo, AWS Glue Studio inicia un nuevo script con texto reutilizable escrito en Python. Si desea escribir un script de Scala en su lugar, primero debe configurar el editor de script para que utilice Scala.

> **Note**
>
> Si elige Scala como lenguaje de programación para el trabajo y utiliza el editor visual para diseñar su trabajo, el script de trabajo generado se escribe en Scala y no es necesario realizar más acciones.

Para escribir un nuevo script de Scala en AWS Glue Studio

1. Cree un nuevo trabajo mediante la opción Spark script editor (Editor de scripts de Spark).
2. En Options (Opciones), elija Create a new script with boilerplate code (Crear un nuevo script con código reutilizable).
3. Elija la pestaña Job details (Detalles del trabajo) y configure el Languaje (Lenguaje) a `Scala` (en lugar de `Python 3`).

   > **Note**
   >
   > La propiedad Type (Tipo) para el trabajo se configura automáticamente en `Spark` cuando elije la opción Spark script editor (Editor de scripts de Spark) para crear un trabajo.

4. Elija la pestaña Script.
5. Elimine el texto reutilizable de Python. Puede reemplazarlo con el siguiente texto reutilizable de Scala.

```
import com.amazonaws.services.glue.{DynamicRecord, GlueContext}
import org.apache.spark.SparkContext
import com.amazonaws.services.glue.util.JsonOptions
import com.amazonaws.services.glue.util.GlueArgParser
import com.amazonaws.services.glue.util.Job

object MyScript {
  def main(args: Array[String]): Unit = {
    val sc: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(sc)

    }
}
```

6. Escriba su script de trabajo de Scala en el editor. Agregue declaraciones `import` adicionales, según sea necesario.

Creación y edición de trabajos de shell de Python en AWS Glue Studio

Cuando elige el editor de scripts de shell de Python para crear un trabajo, puede cargar un script de Python existente o escribir uno nuevo. Si elige escribir un nuevo script, se agrega código reutilizable al nuevo script de trabajo de Python.

Para crear un nuevo trabajo de shell de Python

Consulte las instrucciones en Empezar trabajos en AWS Glue Studio.

Las propiedades de trabajo que se soportan para los trabajos de shell de Python no son las mismas que las soportadas para los trabajos de Spark. La siguiente lista describe los cambios en los parámetros de trabajo disponibles para los trabajos de shell de Python en la pestaña Job details (Detalles del trabajo).

- La propiedad Type (Tipo) para el trabajo se establece automáticamente en `Python Shell` y no se puede cambiar.
- En lugar de Language (Lenguaje), hay una propiedad Python version (Versión de Python) para el trabajo. Actualmente, los trabajos de shell de Python creados en AWS Glue Studio utilizan Python 3.6.
- La propiedad Glue version (Versión de Glue) no está disponible, ya que no se aplica a trabajos de shell de Python.

- En lugar de Worker type (Tipo de empleado) y Number of workers (Número de empleados), se muestra una propiedad Data processing units (Unidades de procesamiento de datos). Esta propiedad de trabajo determina cuántas unidades de procesamiento de datos (DPU) consume el shell de Python al ejecutar el trabajo.
- La propiedad Job bookmark (Marcador de trabajo) no está disponible, porque no se soporta para trabajos de shell de Python.
- En Advanced properties (Propiedades avanzadas), las siguientes propiedades no están disponibles para trabajos de shell de Python.
  - Métricas de trabajo
  - Registro continuo
  - Spark UI (Interfaz de usuario de Spark) y Spark UI logs path (Ruta de los registros de la interfaz de usuario de Spark)
  - Dependent jars path (Ruta de archivos JAR dependientes), en el encabezado Libraries (Bibliotecas)

Cambio de los nodos principales de un nodo en el diagrama de trabajo

Puede cambiar los elementos principales de un nodo para mover nodos dentro del diagrama de trabajo o para cambiar un origen de datos para un nodo.

Para cambiar el nodo principal

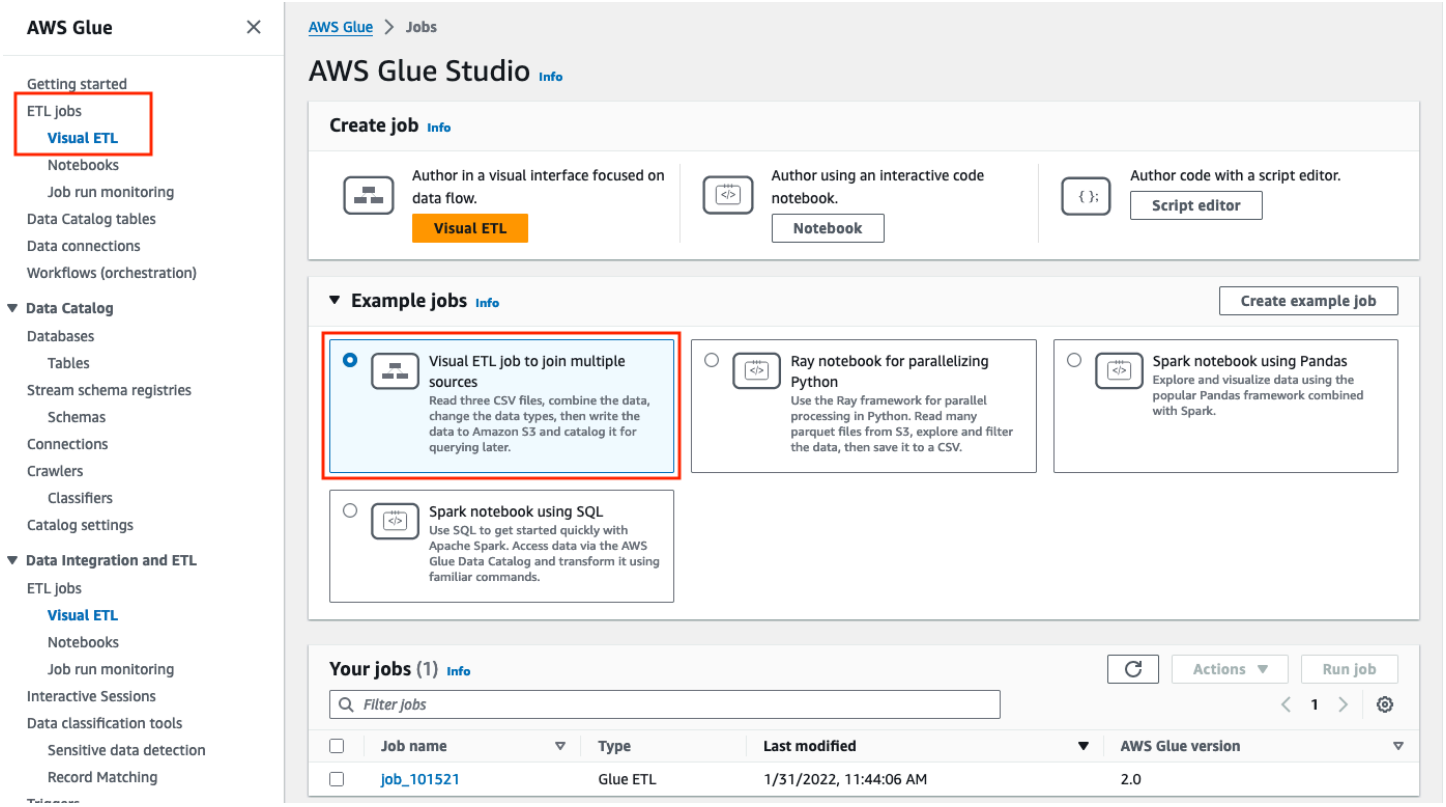
1. Elija el nodo del diagrama de trabajo que desee modificar.
2. En el panel de detalles del nodo, en la pestaña Node properties (Propiedades del nodo), en el encabezado Node parents (Nodos principales) elimine el nodo principal actual.
3. Elija un nuevo nodo principal de la lista.
4. Modifique las demás propiedades del nodo según sea necesario para que coincida con el nodo principal recién seleccionado.

Si modificó un nodo por error, puede utilizar el botón Undo (Deshacer) de la barra de herramientas para revertir la acción.

Eliminación de nodos del diagrama de trabajo

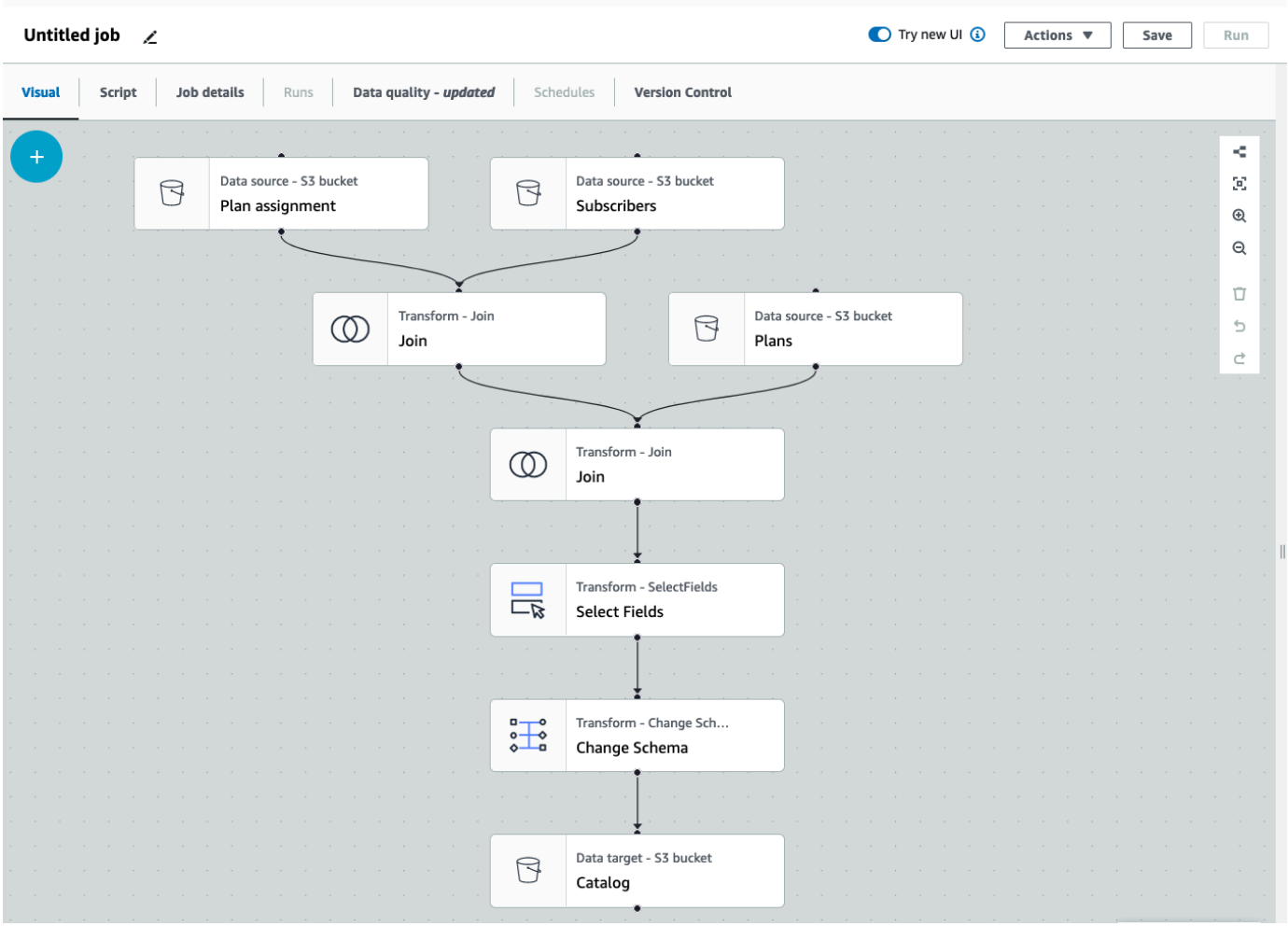
Al realizar trabajos con Visual ETL, puede quitar los nodos del lienzo sin la necesidad de volver a agregar o de reestructurar ningún nodo que esté conectado al nodo quitado.

En el siguiente ejemplo, puede ir a Trabajos de ETL > Visual ETL y, luego, en Ejemplos de trabajos, seleccionar Trabajo de Visual ETL para unir varios orígenes. Seleccione Crear ejemplo de trabajo para crear un trabajo y siga los pasos que se detallan a continuación.



Eliminación de un nodo del lienzo

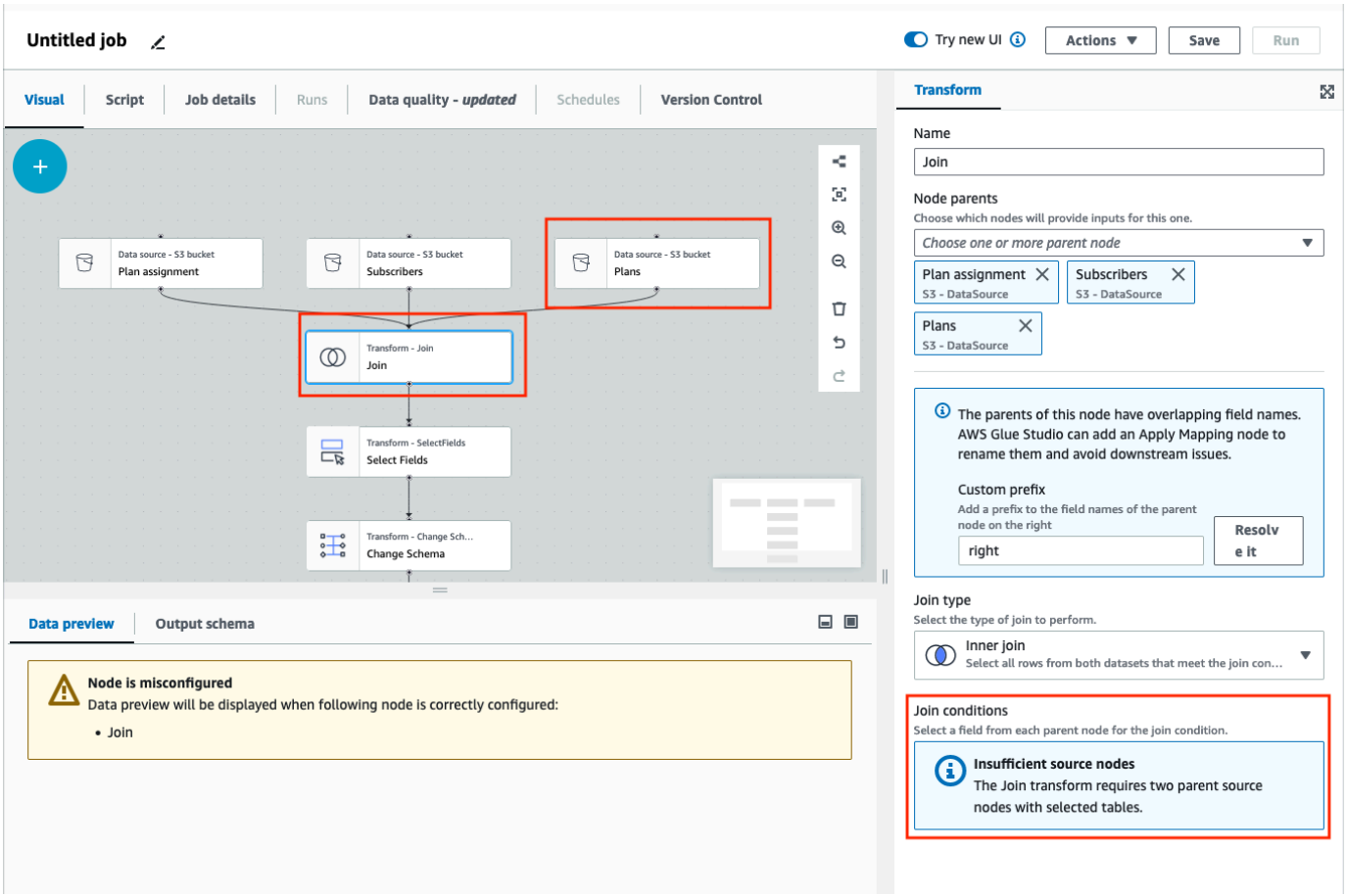
1. Desde la consola de AWS Glue, seleccione Visual ETL en el menú de navegación y seleccione el trabajo existente. El lienzo de trabajo mostrará el trabajo de ejemplo como se muestra a continuación.



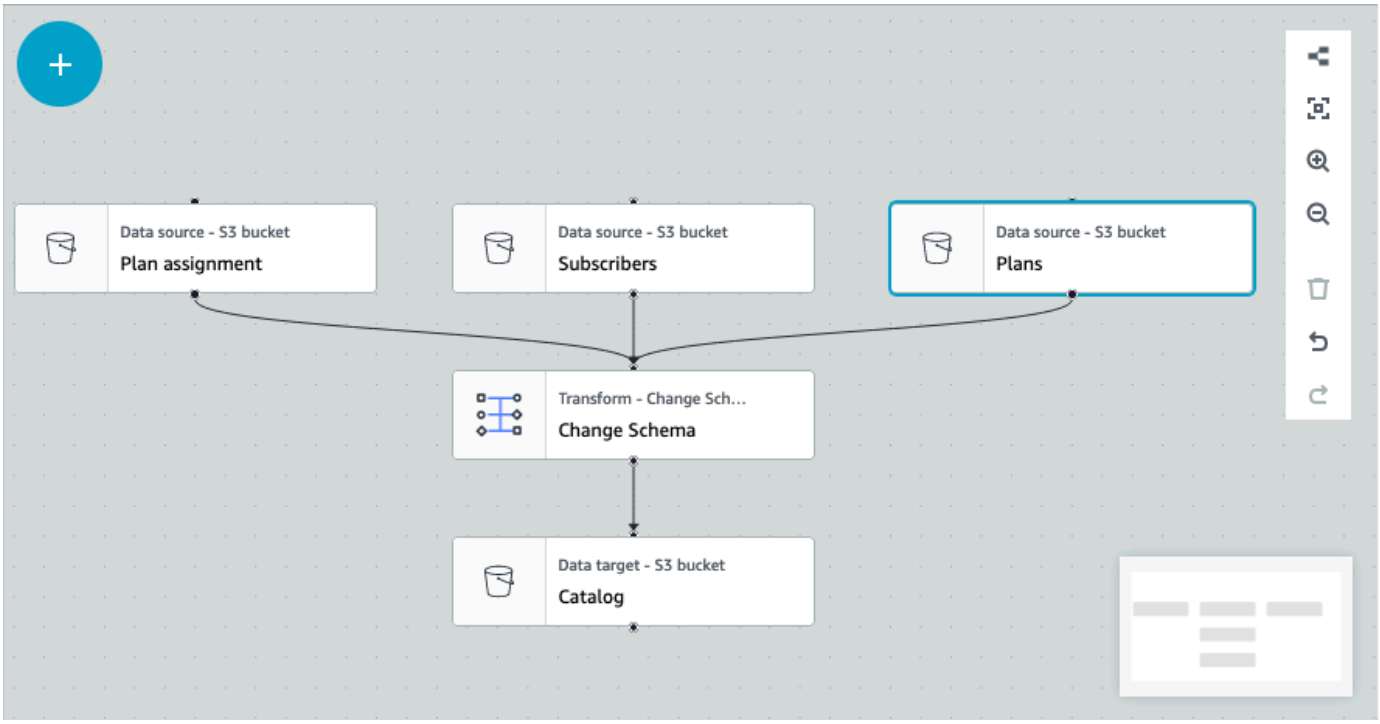
2. Seleccione el nodo que desee quitar. El lienzo hará un acercamiento al nodo. En la barra de herramientas del lado derecho del lienzo, seleccione el ícono Papelera. Así se quitará el nodo y cualquier nodo conectado a ese nodo ocupará su lugar en el flujo de trabajo. En este ejemplo, se eliminó el primer nodo Unión del lienzo.

   Si se elimina un nodo del flujo de trabajo, AWS Glue reorganizará los nodos para que su organización no resulte en un flujo de trabajo no válido. Es posible que necesite corregir la configuración del nodo.

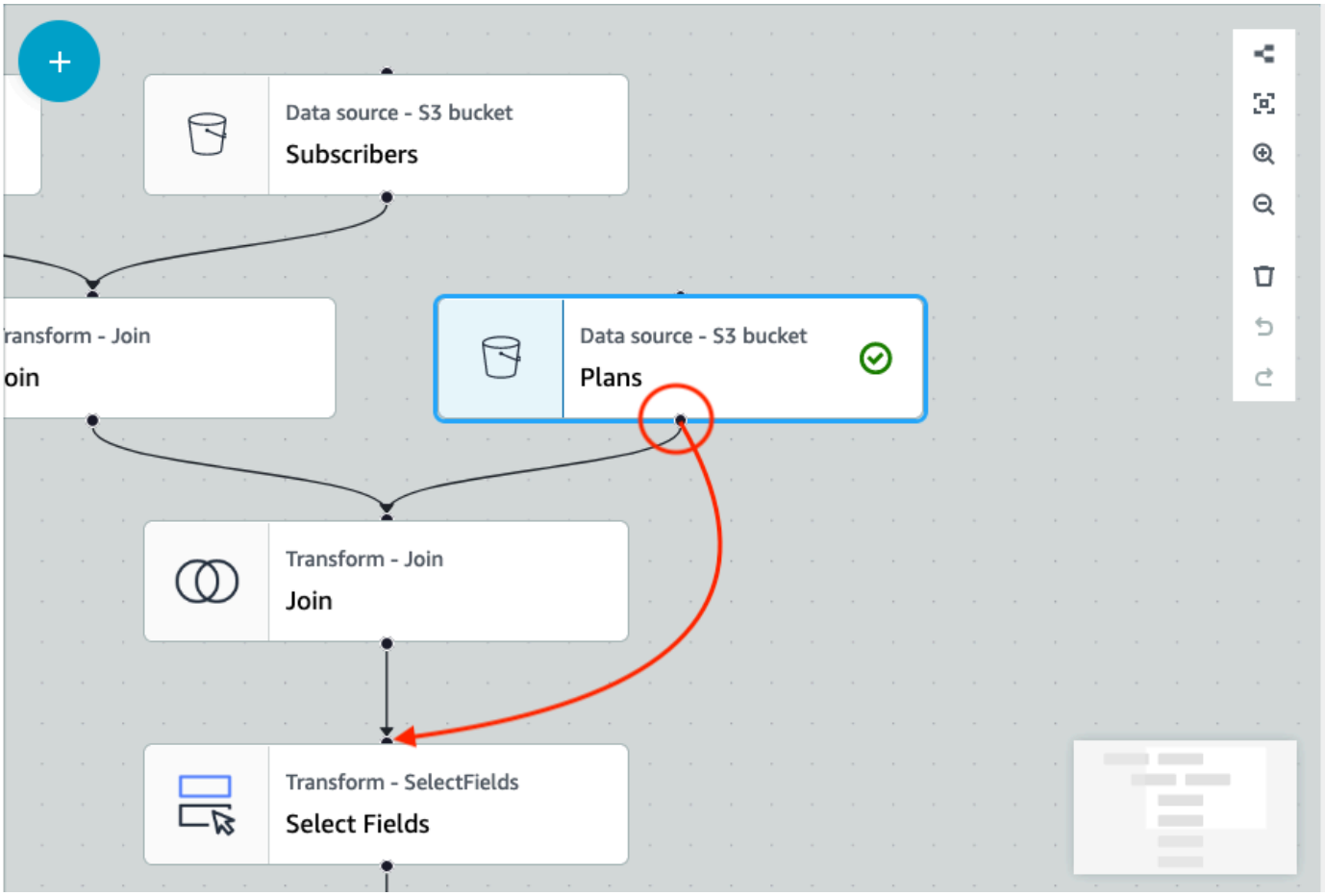
   En el ejemplo, se quitó el nodo Unión que se encontraba debajo del nodo Suscriptores. Como resultado, el nodo de origen Planes se movió al nivel superior y sigue conectado al nodo secundario Unión. El nodo Unión ahora necesita configuración adicional, ya que Unión requiere dos nodos de origen principales con las tablas seleccionadas. En la pestaña Transformar, que se encuentra en el lado derecho del lienzo, se muestran los requisitos que faltan en Condiciones de unión.



3. Elimine el segundo nodo Unión y el nodo Seleccionar campos. Al eliminar estos nodos, el flujo de trabajo se verá como se muestra a continuación.



4. Para modificar las conexiones de los nodos, haga clic en el controlador del nodo y arrastre la conexión a un nodo nuevo. Esto le permitirá eliminar los nodos y reorganizar los nodos en un flujo lógico. En el ejemplo, la flecha roja indica cómo crear una nueva conexión al hacer clic en el controlador del nodo Planes y arrastrar la conexión al nodo Unión.



5. Si necesita deshacer alguna acción, seleccione el ícono Deshacer justo por debajo del ícono Papelera en la barra de herramientas que se encuentra en el costado derecho del lienzo.

Añadir parámetros de origen y destino al nodo AWS Glue del Catálogo de datos

AWS Glue Studio permite parametrizar trabajos visuales. Dado que los nombres de las tablas de catálogo en el entorno de producción y desarrollo pueden ser diferentes, puede definir y seleccionar los parámetros de tiempo de ejecución para las bases de datos y tablas que se ejecutarán cuando se ejecute el trabajo.

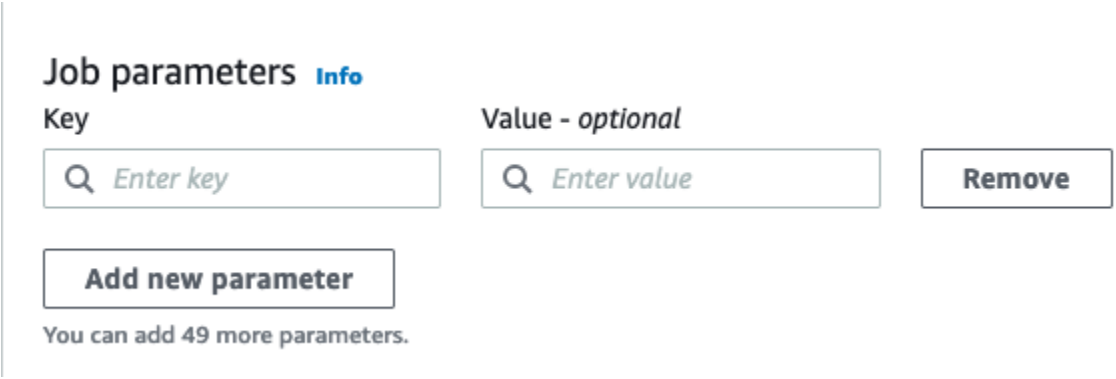
La parametrización del trabajo permite parametrizar los orígenes y destinos, y guardar esos parámetros en el trabajo cuando se utiliza el nodo AWS Glue del Catálogo de datos. Al especificar los orígenes y destinos como parámetros, se habilita la reutilización de los trabajos, especialmente cuando se utiliza el mismo trabajo en varios entornos. Esto es útil para promover el código en los entornos de implementación, ya que le ahorra tiempo y esfuerzo en la administración de los orígenes

y destinos. Además, los parámetros personalizados que especifique anularán cualquier argumento predeterminado para ejecuciones específicas de trabajos de AWS Glue.

Para añadir parámetros de origen y destino

Aunque use el nodo AWS Glue del Catálogo de datos como origen o destino, igualmente puede definir los parámetros de tiempo de ejecución en la sección Propiedades avanzadas en la pestaña Detalles del trabajo.

1. Elija el nodo AWS Glue del Catálogo de datos como nodo de origen o nodo de destino.
2. Elija la pestaña Detalles del trabajo.
3. Elija Propiedades avanzadas.
4. En la sección Parámetros del trabajo, introduzca un valor clave. Por ejemplo, `--db.source` sería el parámetro de un origen de base de datos. Puede introducir cualquier nombre para la clave, siempre y cuando el nombre de esta vaya seguido de un "guión".

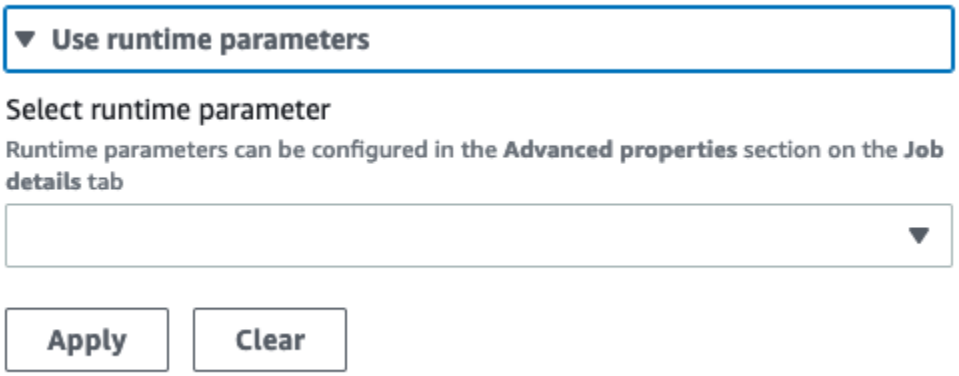


5. Especifique el valor. Por ejemplo, `databasename` sería el valor de la base de datos que se parametriza.
6. Elija Agregar parámetro si quiere agregar más parámetros. Se permite un máximo de 50 parámetros. Una vez definido el par de valor clave, puede utilizar el parámetro en el nodo AWS Glue del Catálogo de datos.

Para seleccionar un parámetro de tiempo de ejecución

> **Note**
>
> El proceso de selección de los parámetros de ejecución para las bases de datos y tablas es el mismo tanto si el nodo AWS Glue del Catálogo de datos es el origen como el destino.

1. Elija el nodo AWS Glue del Catálogo de datos como nodo de origen o nodo de destino.
2. En la pestaña Propiedades del origen de datos - Catálogo de datos, en Base de datos, seleccione Utilizar parámetros en tiempo de ejecución.



3. Elija un parámetro en el menú desplegable. Por ejemplo, cuando seleccione un parámetro definido para una base de datos de origen, la base de datos aparecerá automáticamente en el menú desplegable de la base de datos cuando seleccione Aplicar.
4. En la sección Tabla, elija un parámetro que ya haya definido como tabla de origen. Cuando elige Aplicar, la tabla se rellena automáticamente como la tabla que se va a utilizar.
5. Cuando guarde y ejecute el trabajo, AWS Glue Studio hará referencia a los parámetros seleccionados durante la ejecución del trabajo.

Uso de los sistemas de control de versiones de Git en AWS Glue

> **Note**
>
> Los cuadernos actualmente no son compatibles con el control de versiones en AWS Glue Studio. Sin embargo, el control de versiones es compatible con los scripts de trabajo y trabajos de ETL visuales en AWS Glue.

Si tiene repositorios remotos y quiere administrar los trabajos de AWS Glue mediante sus repositorios, puede usar AWS Glue Studio o AWS CLI para sincronizar los cambios en los repositorios y trabajos en AWS Glue. Cuando sincroniza los cambios de esta manera, llevará el trabajo desde AWS Glue Studio al repositorio o lo extraerá del repositorio a AWS Glue Studio.

Con la integración de Git en AWS Glue Studio, puede:

- Se integra con los sistemas de control de versiones de Git como AWS CodeCommit, GitHub, GitLab y Bitbucket
- Edite trabajos de AWS Glue en AWS Glue Studio si usa trabajos visuales o trabajos de script y los sincroniza con un repositorio
- Parametrice los orígenes y objetivos en los trabajos
- Extraiga trabajos de un repositorio y edítelos en AWS Glue Studio
- Pruebe los trabajos extrayéndolos de las sucursales o enviándolos a estas mediante los flujos de trabajo de varias sucursales en AWS Glue Studio
- Descargue archivos de un repositorio y suba trabajos a AWS Glue Studio para la creación de trabajo entre cuentas
- Utilice la herramienta de automatización de su elección (por ejemplo, Jenkins, AWS CodeDeploy, etc.)

En este video, se muestra cómo puede integrar AWS Glue con Git y crear una canalización de código continua y colaborativa.

Permisos de IAM

Asegúrese de que el trabajo tenga uno de los siguientes permisos de IAM. Para más información sobre cómo configurar los permisos de IAM, consulte Configurar permisos de IAM para AWS Glue Studio.

- `AWSGlueServiceRole`
- `AWSGlueConsoleFullAccess`

Como mínimo, se necesitan las siguientes acciones para la integración de Git:

- `glue:UpdateJobFromSourceControl`: para poder actualizar AWS Glue con un trabajo presente en un sistema de control de versiones
- `glue:UpdateSourceControlFromJob`: para poder actualizar el sistema de control de versiones con un trabajo almacenado en AWS Glue
- `s3:GetObject`: para poder recuperar el script del trabajo mientras se pasa al sistema de control de versiones
- `s3:PutObject`: para poder actualizar el script al extraer un trabajo desde un sistema de control de código de origen

Requisitos previos

Para enviar los trabajos a un repositorio de control de código fuente, necesitará:

- un repositorio que ya haya sido creado por su administrador
- una rama del repositorio
- un token de acceso personal (en el caso de Bitbucket, se trata del token de acceso al repositorio)
- nombre de usuario del propietario del repositorio
- establecer permisos en el repositorio para que AWS Glue Studio pueda leer y escribir en el repositorio
  - GitLab: configure el alcance del token en api, read_repository y write_repository
  - Bitbucket: establezca los permisos para:
    - Pertenencia al espacio de trabajo: leer y escribir
    - Proyectos: escribir, administrador, leer
    - Repositorios: leer, escribir, administrador, eliminar

> **Note**
>
> Cuando se utiliza AWS CodeCommit, no se necesita el token de acceso personal ni el propietario del repositorio. Consulte Introducción a Git y AWS CodeCommit.

Uso de trabajos del repositorio de control de código de origen en AWS Glue Studio

Para extraer un trabajo del repositorio de control de código de origen que no se encuentra en AWS Glue Studio y que usa ese trabajo en AWS Glue Studio, los requisitos previos dependerán del tipo de trabajo.

Para un trabajo visual:

- necesita una carpeta y un archivo JSON de la definición del trabajo que coincida con el nombre del trabajo

  Por ejemplo, consulte la definición de trabajo a continuación. La rama del repositorio debe contener una ruta `my-visual-job/my-visual-job.json` donde tanto la carpeta como el archivo JSON coincidan con el nombre del trabajo

```
{
  "name" : "my-visual-job",
  "description" : "",
  "role" : "arn:aws:iam::aws_account_id:role/Rolename",
  "command" : {
    "name" : "glueetl",
    "scriptLocation" : "s3://foldername/scripts/my-visual-job.py",
    "pythonVersion" : "3"
  },
  "codeGenConfigurationNodes" : "{\"node-nodeID\":{\"S3CsvSource\":
{\"AdditionalOptions\":{\"EnableSamplePath\":false,\"SamplePath\":\"s3://notebook-
test-input/netflix_titles.csv\"},\"Escaper\":\"\",\"Exclusions\":[],\"Name\":\"Amazon
 S3\",\"OptimizePerformance\":false,\"OutputSchemas\":[{\"Columns\":[{\"Name\":
\"show_id\",\"Type\":\"string\"},{\"Name\":\"type\",\"Type\":\"string\"},{\"Name\":
\"title\",\"Type\":\"choice\"},{\"Name\":\"director\",\"Type\":\"string\"},{\"Name\":
\"cast\",\"Type\":\"string\"},{\"Name\":\"country\",\"Type\":\"string\"},{\"Name\":
\"date_added\",\"Type\":\"string\"},{\"Name\":\"release_year\",\"Type\":\"bigint\"},
{\"Name\":\"rating\",\"Type\":\"string\"},{\"Name\":\"duration\",\"Type\":\"string
\"},{\"Name\":\"listed_in\",\"Type\":\"string\"},{\"Name\":\"description\",\"Type
\":\"string\"}]}],\"Paths\":[\"s3://dalamgir-notebook-test-input/netflix_titles.csv
\"],\"QuoteChar\":\"quote\",\"Recurse\":true,\"Separator\":\"comma\",\"WithHeader
\":true}}}"
}
```

Para un trabajo de script:

- necesita una carpeta, un archivo JSON de la definición del trabajo y el script
- la carpeta y el archivo JSON deben coincidir con el nombre del trabajo. El nombre del script debe coincidir con el valor de `scriptLocation` en la definición del trabajo junto con la extensión del archivo

  Por ejemplo, en la definición de trabajo que aparece a continuación, la rama del repositorio debe contener una ruta `my-script-job/my-script-job.json` y `my-script-job/my-script-job.py`. El nombre del script debe coincidir con el nombre que aparece en el valor `scriptLocation`, incluida la extensión del script

```
{
  "name" : "my-script-job",
  "description" : "",
```

```
  "role" : "arn:aws:iam::aws_account_id:role/Rolename",
  "command" : {
    "name" : "glueetl",
    "scriptLocation" : "s3://foldername/scripts/my-script-job.py",
    "pythonVersion" : "3"
  }
}
```

Limitaciones

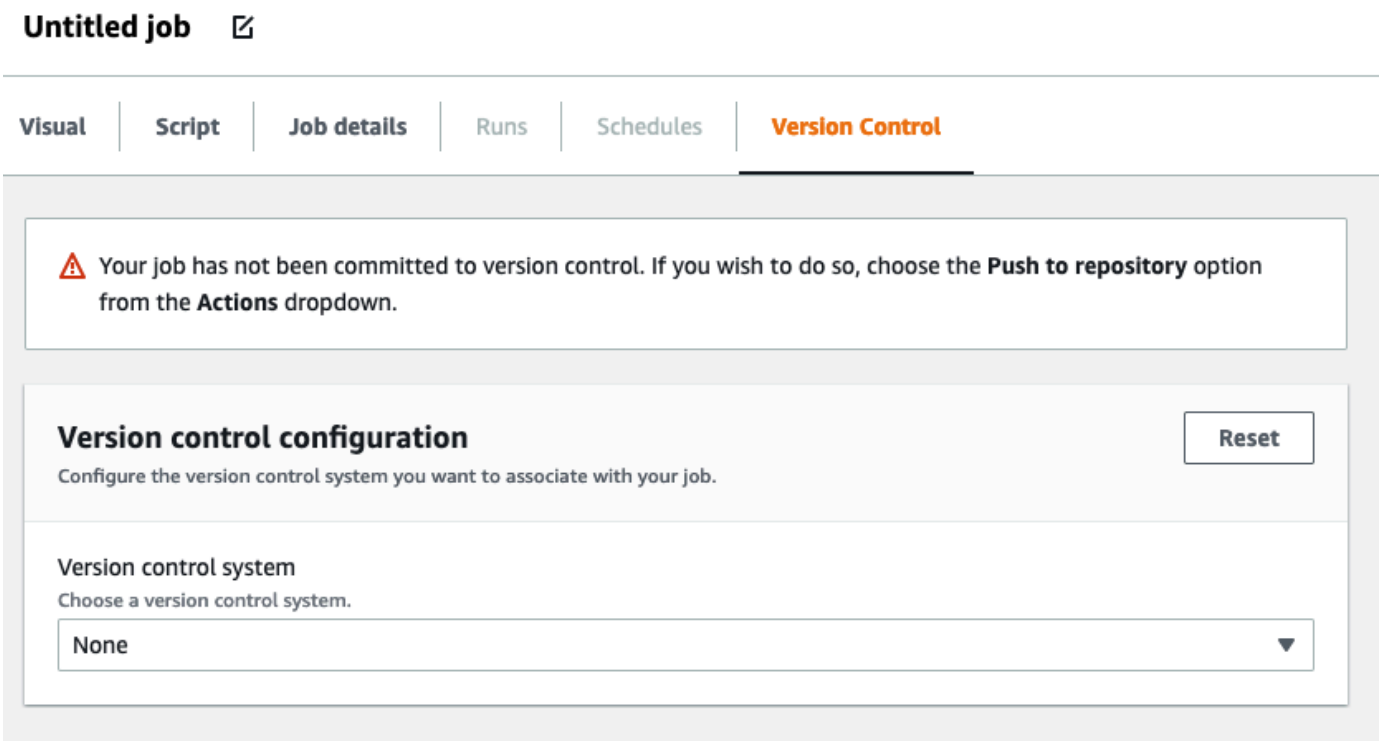
- En la actualidad, AWS Glue no es compatible con las extracciones ni los envíos a GitLab-Groups.

Conectar repositorios de control de versiones con AWS Glue

Puede introducir los detalles de su repositorio de control de versiones y administrarlos en la pestaña Control de versiones en el editor de trabajo de AWS Glue Studio. Para integrarlo con su repositorio de Git, debe conectarse al repositorio cada vez que inicie sesión en AWS Glue Studio.

Para conectar un sistema de control de versiones de Git:

1. En AWS Glue Studio, comience un nuevo trabajo y elija la pestaña Control de versiones.



2. En el Sistema de control de versiones, elija el servicio Git entre las opciones disponibles mediante un clic en el menú desplegable.

   - AWS CodeCommit
   - GitHub
   - GitLab
   - Bitbucket

3. En función del sistema de control de versiones Git que elija, tendrá que completar diferentes campos.

   Para AWS CodeCommit:

   Complete la configuración del repositorio seleccionando el repositorio y la rama para su trabajo:

   - Repositorio: si ha configurado repositorios en AWS CodeCommit, seleccione el repositorio en el menú desplegable. Los repositorios se rellenarán automáticamente en la lista
   - Rama: seleccione la rama en el menú desplegable
   - Carpeta: opcional: escriba el nombre de la carpeta en la que quiere guardar el trabajo. Si se deja vacía, se crea automáticamente una carpeta. El nombre de la carpeta es el nombre del trabajo de forma predeterminada.

   Para GitHub:

   Complete la configuración de GitHub con los campos:

   - Token de acceso: este es el token que proporciona el repositorio de GitHub. Para más información sobre los tokens de acceso personal, consulte Documentos de GitHub
   - Propietario del repositorio: es el propietario del repositorio de GitHub.

   Complete la configuración del repositorio con la selección de este y la rama de GitHub.

   - Repositorio: si ha configurado repositorios en GitHub, seleccione el repositorio en el menú desplegable. Los repositorios se rellenarán automáticamente en la lista
   - Rama: seleccione la rama en el menú desplegable

- Carpeta: opcional: escriba el nombre de la carpeta en la que quiere guardar el trabajo. Si se deja vacía, se crea automáticamente una carpeta. El nombre de la carpeta es el nombre del trabajo de forma predeterminada.

Para GitLab:

> **Note**
>
> En la actualidad, AWS Glue no es compatible con las extracciones ni los envíos a GitLab-Groups.

- Token de acceso: este es el token que proporciona el repositorio de GitLab. Para más información sobre los tokens de acceso personal, consulte Tokens de acceso personal de GitLab
- Propietario del repositorio: es el propietario del repositorio de GitLab.

Complete la configuración del repositorio con la selección de este y la rama de GitLab.

- Repositorio: si ha configurado repositorios en GitLab, seleccione el repositorio en el menú desplegable. Los repositorios se rellenarán automáticamente en la lista
- Rama: seleccione la rama en el menú desplegable
- Carpeta: opcional: escriba el nombre de la carpeta en la que quiere guardar el trabajo. Si se deja vacía, se crea automáticamente una carpeta. El nombre de la carpeta es el nombre del trabajo de forma predeterminada.

Para Bitbucket:

- Contraseña de la aplicación: Bitbucket no utiliza tokens de acceso a repositorios, sino contraseñas de aplicaciones. Para obtener más información sobre las contraseñas de aplicaciones, consulte Contraseñas de aplicaciones.
- Propietario del repositorio: es el propietario del repositorio de Bitbucket. En Bitbucket, el propietario es el creador del repositorio.

Seleccione el espacio de trabajo, el repositorio, la rama y la carpeta desde Bitbucket para completar la configuración del repositorio.

- Espacio de trabajo: si tiene espacios de trabajo configurados en Bitbucket, seleccione el espacio de trabajo en el menú desplegable. Los espacios de trabajo se rellenan automáticamente
- Repositorio: si ha configurado repositorios en Bitbucket, seleccione el repositorio en el menú desplegable. Los repositorios se rellenan automáticamente
- Rama: seleccione la rama en el menú desplegable. Las ramas se rellenan automáticamente
- Carpeta: opcional: escriba el nombre de la carpeta en la que quiere guardar el trabajo. Si se deja vacío, se creará automáticamente una carpeta con el nombre del trabajo.

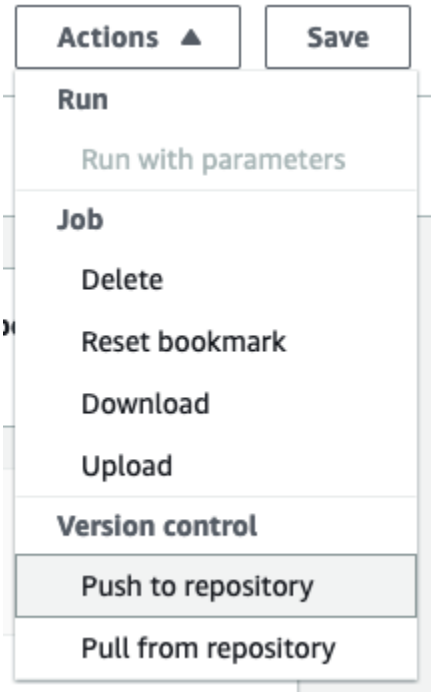
4. Seleccione Guardar en la parte superior del trabajo de AWS Glue Studio

Envío de trabajos de AWS Glue al repositorio de origen

Una vez que haya introducido los detalles de su sistema de control de versiones, puede editar los trabajos en AWS Glue Studio y empujar los trabajos a su repositorio de origen. Si no está familiarizado con los conceptos de Git, como empujar y tirar, consulta este tutorial sobre Introducción a Git y AWS CodeCommit.

Para enviar su trabajo a un repositorio, debe introducir los detalles de su sistema de control de versiones y guardar el trabajo.

1. En el trabajo AWS Glue Studio, elija Acciones. Esto abrirá opciones de menú adicionales.



2. Elija Insertar en el repositorio.

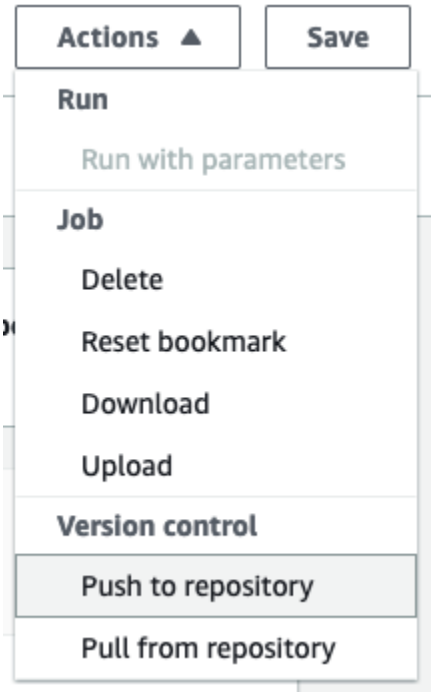
   Esta acción guardará el trabajo. Cuando se pasa al repositorio, AWS Glue Studio empuja el último cambio guardado. Si el mismo usuario u otro usuario modificó el trabajo en el repositorio y no está sincronizado con el trabajo en AWS Glue Studio, el trabajo del repositorio se sobrescribe con el trabajo guardado en AWS Glue Studio cuando se empuja el trabajo desde AWS Glue Studio.

3. Elija Confirmar para completar la acción. Esto crea una nueva confirmación en el repositorio. Si está utilizando AWS CodeCommit, un mensaje de confirmación mostrará un enlace a la última confirmación en AWS CodeCommit.

Sacar trabajos de AWS Glue del repositorio de origen

Una vez que haya introducido los detalles de su repositorio de Git en la pestaña Control de versiones, también puede extraer trabajos de su repositorio y editarlos en AWS Glue Studio.

1. En el trabajo de AWS Glue Studio, elija Acciones. Esto abrirá opciones de menú adicionales.



2. Elija Extraer del repositorio.
3. Seleccione Confirmar. Esto toma la confirmación más reciente del repositorio y actualiza el trabajo en AWS Glue Studio.
4. Edite su trabajo en AWS Glue Studio. Si realiza cambios, puede sincronizar su trabajo con el repositorio al elegir Insertar en el repositorio desde el menú desplegable Acciones.

Creación de código con cuadernos de AWS Glue Studio

Los ingenieros de datos pueden crear trabajos de AWS Glue de forma más rápida y sencilla que antes mediante el uso de la interfaz de cuaderno interactiva en AWS Glue Studio o sesiones interactivas en AWS Glue.

Temas

- Información general sobre el uso de cuaderno
- Creación de un trabajo de ETL mediante cuaderno en AWS Glue Studio
- Componentes del editor de cuaderno
- Cómo guardar el cuaderno y el script de trabajo
- Administración de las sesiones de cuaderno
- Uso CodeWhisperer con AWS Glue Studio notebooks

Información general sobre el uso de cuaderno

AWS Glue Studio permite crear trabajos de forma interactiva en una interfaz de cuaderno basada en cuadernos de Jupyter. A través de los cuaderno en AWS Glue Studio, es posible editar scripts de trabajos y el código de integración de datos y ver el resultado sin que sea necesario ejecutar un trabajo completo. También es posible agregar un marcado y guardar cuaderno como archivos .ipynb y scripts de trabajo. Puede iniciar un cuaderno sin instalar software en forma local ni administrar servidores. Una vez que esté satisfecho con el código, AWS Glue Studio puede convertir el cuaderno en un trabajo de Glue con solo hacer clic en un botón.

Algunos de los beneficios de utilizar cuaderno son:

- No hay clúster que aprovisionar o administrar
- No hay que pagar por clústeres inactivos
- No se requiere una configuración inicial
- No se requiere instalación de cuadernos de Jupyter
- Mismo tiempo de ejecución y plataforma que ETL de AWS Glue

Al iniciar un cuaderno a través de AWS Glue Studio, todos los pasos de configuración ya han sido completados para que, apenas después de unos segundos, pueda explorar los datos y comenzar a desarrollar el script de trabajo. AWS Glue Studio configura un cuaderno de Jupyter con el kernel de Jupyter de AWS Glue. No es necesario configurar VPC, conexiones de red ni puntos de conexión de desarrollo para utilizar este cuaderno.

Para crear trabajos mediante la interfaz de cuaderno:

- Configure los permisos de IAM necesarios.
- Inicie una sesión de cuaderno para crear un trabajo.
- Escriba código en las celdas en el cuaderno.
- Ejecute y pruebe el código para ver el resultado.
- Guarde el trabajo.

Una vez guardado, el cuaderno es un trabajo completo de AWS Glue. Puede administrar todos los aspectos del trabajo, tales como la programación de ejecuciones de trabajos, la configuración de parámetros del trabajo y la visualización del historial de ejecuciones de trabajos justo al lado del cuaderno.

Creación de un trabajo de ETL mediante cuaderno en AWS Glue Studio

Para empezar a utilizar los cuaderno en la consola de AWS Glue Studio

1. Adjunte políticas de AWS Identity and Access Management al usuario de AWS Glue Studio y cree un rol de IAM para el trabajo y cuaderno de ETL.
2. Configure una seguridad adicional de IAM para cuadernos, como se describe en Concesión de permisos para el rol de IAM.
3. Abra la consola de AWS Glue Studio en https://console.aws.amazon.com/gluestudio/.

   > **Note**
   >
   > Compruebe que el navegador no bloquea las cookies de terceros. Cualquier navegador que bloquee las cookies de terceros, ya sea de manera predeterminada o porque el usuario lo haya configurado así, impedirá que se inicien los cuadernos. Para obtener más información sobre cómo administrar las cookies, consulte:

   - Chrome
   - Firefox
   - Safari
4. Elija el icono Jobs (Trabajos) en el menú de navegación de la izquierda.
5. Elija Jupyter notebook (Cuaderno de Jupyter) y, luego, elija Create (Crear) para iniciar una nueva sesión de cuaderno.
6. En la página Create job in Jupyter notebook (Crear trabajo en cuaderno de Jupyter), proporcione el nombre del trabajo y elija qué rol de IAM desea utilizar. Seleccione Crear trabajo.

   Tras un breve momento, aparece el editor de cuaderno.
7. Después de agregar el código, debe ejecutar la celda para iniciar una sesión. La celda se puede ejecutar de varias formas:

   - Pulse el botón de reproducción.
   - Use un método abreviado de teclado:
     - En MacOS, Command (Comando) + Enter (Intro) para ejecutar la celda.
     - En Windows, Shift + Enter (Intro) para ejecutar la celda.

Para obtener información sobre cómo escribir código mediante una interfaz de cuaderno de Jupyter, consulte The Jupyter Notebook User Documentation (Documentación del usuario de cuaderno de Jupyter).

8. Para probar el script, ejecute el script completo o celdas individuales. Cualquier salida de comando se mostrará en el área situada debajo de la celda.
9. Una vez que haya terminado de desarrollar el cuaderno, puede guardar el trabajo y luego ejecutarlo. Encontrará el script en la pestaña Script. Cualquier comando mágico que haya agregado al cuaderno se eliminará y no se guardará como parte del script del trabajo de AWS Glue generado. AWS Glue Studio agregará automáticamente un `job.commit()` al final del script generado desde el contenido del cuaderno.

   Para obtener más información sobre cómo ejecutar un trabajo, consulte Iniciar una ejecución de trabajo.

Componentes del editor de cuaderno

La interfaz del editor de cuaderno tiene las siguientes secciones principales.

- Interfaz de cuaderno (panel principal) y barra de herramientas
- Pestañas de edición de trabajo

Editor de cuaderno

El editor de cuaderno de AWS Glue Studio se basa en la aplicación de cuaderno de Jupyter. La interfaz de cuaderno de AWS Glue Studio es similar a la provista por Bloc de notas de Jupyter, la cual se describe en la sección Notebook user interface (Interfaz de usuario de cuaderno). El cuaderno utilizado por las sesiones interactivas es un cuaderno de Jupyter.

Aunque el cuaderno de AWS Glue Studio es similar a cuaderno de Jupyter, difiere en algunos aspectos clave:

- en la actualidad, el cuaderno de AWS Glue Studio no puede instalar extensiones
- no se pueden utilizar varias pestañas; existe una relación 1:1 entre un trabajo y un cuaderno
- el cuaderno de AWS Glue Studio no tiene el mismo menú de archivos principales que existe en los cuadernos de Jupyter

- actualmente, el cuaderno de AWS Glue Studio solo se ejecuta con el kernel de AWS Glue. Tenga en cuenta que no puede actualizar el kernel por su cuenta.

pestañas de edición de trabajo de AWS Glue Studio

Las pestañas que utiliza para interactuar con el trabajo de ETL se encuentran en la parte superior de la página del cuaderno. Son similares a las pestañas que aparecen en el editor visual de trabajos de AWS Glue Studio y realizan las mismas acciones.

- Notebook (Cuaderno): utilice esta pestaña para ver el script de trabajo mediante la interfaz del cuaderno.
- Job details (Detalles del trabajo): configure el entorno y las propiedades de las ejecuciones de trabajos.
- Runs (Ejecuciones): permite ver información sobre las ejecuciones anteriores de este trabajo.
- Schedules (Cronograma): configure un cronograma para ejecutar su trabajo en momentos específicos.

Cómo guardar el cuaderno y el script de trabajo

Puede guardar el cuaderno y el script de trabajo que esté creando en cualquier momento. Simplemente elija el botón Save (Guardar) en la esquina superior derecha, igual que si estuviera en el editor visual o de script.

Cuando elige Save (Guardar), el archivo de cuaderno se guarda en las ubicaciones predeterminadas:

- De forma predeterminada, el script de trabajo se guarda en la ubicación de Amazon S3 indicada en la pestaña Job Details (Detalles del trabajo), en Advanced properties (Propiedades avanzadas), en la propiedad de detalles del trabajo Script path (Ruta de script). Los scripts de trabajo se guardan en una subcarpeta llamada `Scripts`.
- De forma predeterminada, el archivo de cuaderno (`.ipynb`) se guarda en la ubicación de Amazon S3 indicada en la pestaña Job Details (Detalles del trabajo), en Advanced Properties (Propiedades avanzadas), en Script path (Ruta de script) en los detalles del trabajo. Los archivos del cuaderno se guardan en una subcarpeta llamada`Notebooks`.

> **Note**
>
> Al guardar el trabajo, el script de trabajo contiene solo las celdas de código del cuaderno. Las celdas y las magias de Markdown no se incluyen en el script de trabajo. Sin embargo, el archivo `.ipynb` contendrá cualquier markdown y magia.

Después de guardar el trabajo, puede ejecutarlo mediante el script que creó en el cuaderno.

Administración de las sesiones de cuaderno

Los cuaderno en AWS Glue Studio se basan en la característica de sesiones interactivas de AWS Glue. El uso de sesiones interactivas conlleva un costo. Para ayudar a administrar los costos, puede monitorear las sesiones creadas para su cuenta y ajustar la configuración predeterminada de todas las sesiones.

Cambio del tiempo de espera predeterminado de todas las sesiones de cuaderno

De manera predeterminada, el cuaderno de AWS Glue Studio aprovisionado agota el tiempo de espera al cabo de 12 horas si se inicia y no se ejecuta ninguna celda. No lleva asociado ningún coste y el tiempo de espera no se puede configurar.

Una vez que se ejecuta una celda, se inicia una sesión interactiva. Esta sesión tiene un tiempo de espera predeterminado de 48 horas. Este tiempo de espera se puede configurar ejecutando un comando mágico `%idle_timeout` antes de ejecutar una celda.

Para modificar el tiempo de espera de sesión predeterminado para los cuaderno en AWS Glue Studio

1. En el cuaderno, ingrese el comando mágico `%idle_timeout` en una celda y especifique el valor de tiempo de espera en minutos.
2. Por ejemplo: `%idle_timeout 15` cambiará el tiempo de espera predeterminado a 15 minutos. Si la sesión no se utiliza en 15 minutos, se detiene automáticamente.

Instalación de módulos adicionales de Python

Si desea instalar módulos adicionales a su sesión mediante pip, puede hacerlo con `%additional_python_modules` para agregarlos a la sesión:

```
%additional_python_modules awswrangler, s3://mybucket/mymodule.whl
```

Todos los argumentos para additional_python_modules se pasan a `pip3 install -m <>`.

Para ver la lista de los módulos de Python disponibles, consulte Uso de bibliotecas Python con AWS Glue.

Cambio de la configuración de AWS Glue

Se pueden utilizar comandos mágicos para controlar los valores de configuración de los trabajos de AWS Glue. Si desea cambiar el valor de configuración de un trabajo, debe utilizar el comando mágico adecuado en el cuaderno. Consulte Comandos mágicos compatibles con las sesiones interactivas de AWS Glue para Jupyter.

> **Note**
>
> Ya no está disponible la función de anular las propiedades de una sesión en ejecución. Si desea cambiar las configuraciones de la sesión, debe detenerla, establecer las nuevas configuraciones y, a continuación, iniciar una nueva sesión.

AWS Glue admite varios tipos de empleados. Puede configurar el tipo de empleado con `%worker_type`. Por ejemplo: `%worker_type G.2X` . El valor predeterminado es G.1X.

También puede especificar el número de empleados con `%number_of_workers`. Por ejemplo, para especificar 40 empleados: `%number_of_workers 40`.

Para obtener más información, consulte Defining Job Properties (Definición de las propiedades del trabajo).

Cómo detener una sesión de cuaderno

Para detener una sesión de cuaderno, utilice el comando mágico `%stop_session`.

Si se aleja del cuaderno en la consola de AWS, recibirá un mensaje de advertencia en el que podrá elegir detener la sesión.

Uso CodeWhisperer con AWS Glue Studio notebooks

AWS Glue Studio permite crear trabajos de forma interactiva en una interfaz de cuaderno basada en cuadernos de Jupyter. Su uso CodeWhisperer mejora la experiencia de creación en las libretas. AWS Glue Studio

La CodeWhisperer extensión Amazon permite escribir código mediante la generación de recomendaciones de código y la sugerencia de mejoras relacionadas con los problemas de código.

¿Qué es Amazon CodeWhisperer?

Amazon CodeWhisperer es un servicio impulsado por el aprendizaje automático que ayuda a mejorar la productividad de los desarrolladores. CodeWhisperer Para ello, genera recomendaciones de código basadas en los comentarios de los desarrolladores en lenguaje natural y en su código en el IDE. Durante la versión preliminar, Amazon estará CodeWhisperer disponible para Java, Python JavaScript, C# y los lenguajes de TypeScript programación. El servicio se integra con Amazon SageMaker Studio JupyterLab, instancias de Amazon SageMaker notebook y otros entornos de desarrollo integrados (IDE).

Para obtener más información, consulta la sección Configuración CodeWhisperer con AWS Glue Studio.

Estados de ejecución de tareas de AWS Glue en la consola

Puede ver el estado de un trabajo de extracción, transformación y carga (ETL) de AWS Glue mientras se está ejecutando o después de su detención. Puede ver el estado mediante la consola de AWS Glue. Para obtener más información acerca de los estados de ejecución de un trabajo, consulte the section called “Estados de ejecuciones de trabajos”.

Acceso al panel de monitoreo de trabajos

Puede acceder al panel de monitoreo de trabajos al seleccionar el enlace Monitoreo en el panel de navegación de AWS Glue.

Información general del panel de monitoreo de trabajos

El panel de monitoreo de trabajos proporciona un resumen general de las ejecuciones de trabajos, con totales para los trabajos con un estado de En ejecución, Cancelado, Éxito o Error. Los mosaicos

adicionales proporcionan la tasa general de éxito de ejecución del trabajo, el uso estimado de DPU para los trabajos, un desglose de los recuentos de estado del trabajo por tipo de trabajo, tipo de empleado y día.

Los gráficos de los mosaicos son interactivos. Puede elegir cualquier bloque de un gráfico para ejecutar un filtro que muestre sólo esos trabajos en la tabla Ejecuciones de trabajo de la parte inferior de la página.

Puede cambiar el intervalo de fechas de la información mostrada en esta página mediante el selector Intervalo de fechas. Al cambiar el intervalo de fechas, los mosaicos de información se ajustan para mostrar los valores según la cantidad especificada de días antes de la fecha actual. También puede utilizar un intervalo de fechas específico si elige Personalizado desde el selector de intervalo de fechas.

Vista de las ejecuciones de trabajo

> **Note**
>
> El historial de ejecución de trabajos está disponible durante 90 días para su flujo de trabajo y ejecución de trabajos.

La lista de recursos Ejecuciones de trabajo muestra los trabajos para el intervalo de fechas especificado y los filtros.

Puede filtrar los trabajos según criterios adicionales, como el estado, el tipo de empleado, el tipo de trabajo y el nombre del trabajo. En el cuadro de filtro situado en la parte superior de la tabla, puede introducir el texto que desea utilizar como filtro. Los resultados de la tabla se actualizan con filas que contienen texto coincidente a medida que se escribe el texto.

Puede ver un subconjunto de los trabajos cuando selecciona elementos de los gráficos del panel de monitoreo de trabajos. Por ejemplo, si elige el número de trabajos en ejecución en el mosaico Resumen de ejecuciones de trabajo, la lista Ejecuciones de trabajo muestra sólo los trabajos que actualmente tienen un estado de `Running`. Si elige una de las barras del gráfico de barras Desglose por tipo de empleado, solo se muestran las ejecuciones de trabajos con el tipo y el estado de empleado coincidentes en la lista Ejecuciones de trabajo.

La lista de recursos Ejecuciones de trabajo muestra los detalles del trabajo. Puede ordenar las filas de la tabla si elige un encabezado de columna. Esta tabla contiene la siguiente información:

Propiedad	Descripción
Nombre de trabajo	El nombre del trabajo.
Tipo	El tipo del entorno de trabajo: • Glue ETL: se ejecuta en un entorno Apache Spark administrado por AWS Glue. • Glue Streaming: se ejecuta en un entorno Apache Spark y realiza ETL en flujos de datos. • Shell de Python: ejecuta scripts de Python como intérprete de comandos.
Hora de inicio	La fecha y la hora en que se inició la ejecución de este flujo de trabajo.
Hora de finalización	La fecha y la hora en que se completó la ejecución de este trabajo.
Estado de ejecución	El estado actual de la ejecución de flujo de trabajo. Los valores pueden ser: • `STARTING` • `RUNNING` • `STOPPING` • `STOPPED` • `SUCCEEDED` • `FAILED` • `TIMEOUT`
Tiempo de ejecución	El periodo de tiempo que la ejecución de flujo de trabajo consumió recursos.
Capacidad	El número de unidades de procesamiento de datos (DPU) de AWS Glue asignadas a esta ejecución de trabajo. Para obtener más

Propiedad	Descripción
	información acerca de la planificación de capacidad, consulte Monitoreo para planificar la capacidad de DPU en la Guía para desarrolladores de AWS Glue.

Propiedad	Descripción
Tipo de empleado	El tipo de empleado predefinido que se asigna cuando se ejecuta un trabajo. Los valores pueden ser `G.1X`, `G.2X`, `G.4X` o `G.8X`. • **`G.1X`**: al elegir este tipo, también debe proporcionar un valor para Número de empleados. Cada trabajador se asigna a 1 DPU (4 vCPU, 16 GB de memoria) con un disco de 84 GB (aproximadamente 34 GB libres). Le recomendamos este tipo de proceso de trabajo para trabajos con un uso intensivo de la memoria. Esta es la opción predeterminada de Tipo de empleado para trabajos de AWS Glue versión 2.0 o posterior . • **`G.2X`**: al elegir este tipo, también debe proporcionar un valor para Number of workers (Número de empleados). Cada trabajador se asigna a 2 DPU (8 vCPU, 32 GB de memoria) con un disco de 128 GB (aproximadamente 77 GB libres). Recomendamos este tipo de empleado para trabajos con una utilización intensiva de la memoria y trabajos que ejecuten transform aciones de machine learning. • **`G.4X`**: al elegir este tipo, también debe proporcionar un valor para Número de empleados. Cada trabajador se asigna a 4 DPU (16 vCPU, 64 GB de memoria) con un disco de 256 GB (aproximadamente 235 GB libres). Recomendamos este tipo de trabajador para los trabajos cuyas cargas de trabajo contengan las transformaciones, agregaciones, uniones y consultas más

Propiedad	Descripción
	exigentes. Este tipo de trabajador solo está disponible para los trabajos de Spark ETL de la versión 3.0 de AWS Glue o posterior es en las siguientes regiones de AWS: Este de EE. UU. (Ohio), Este de EE. UU. (Norte de Virginia), Oeste de EE. UU. (Oregón), Asia Pacífico (Singapur), Asia Pacífico (Sídney), Asia Pacífico (Tokio), Canadá (centro), Europa (Fráncfort), Europa (Irlanda) y Europa (Estocolmo). • **`G.8X`**: al elegir este tipo, también debe proporcionar un valor para Número de empleados. Cada trabajador se asigna a 8 DPU (32 vCPU, 128 GB de memoria) con un disco de 512 GB (aproximadamente 487 GB libres). Recomendamos este tipo de trabajador para los trabajos cuyas cargas de trabajo contengan las transformaciones, agregaciones, uniones y consultas más exigentes. Este tipo de trabajador solo está disponible para los trabajos ETL de Spark de la versión 3.0 de AWS Glue o posteriores, en las mismas regiones de AWS compatibles con el tipo de trabajador `G.4X`.
Horas de DPU	El número estimado de DPU utilizadas para la ejecución de trabajo. Una DPU es una medida relativa de la potencia de procesamiento. Las DPU se utilizan para determinar el costo de ejecutar su trabajo. Para obtener más informaci ón, consulte la página de precios de AWS Glue.

Puede elegir cualquier ejecución de trabajo de la lista y ver información adicional. Elija una ejecución de trabajo y luego realice una de las operaciones siguientes:

- Elija el menú Acciones y la opción Ver trabajo para ver el trabajo en el editor visual.
- Elija el menú Acciones y la opción Detener ejecución para detener la ejecución actual del trabajo.
- Elija el botón Ver registros de CloudWatch para ver los registros de ejecución de trabajo para ese trabajo.
- Elija Ver detalles para ver la página de detalles de ejecución de trabajo.

Visualización de los registros de ejecución de trabajo

Puede ver los registros de trabajo de diversas formas:

- En la página Monitoreo, en la tabla Ejecuciones de trabajo, elija una ejecución de trabajo y, a continuación, elija Ver registros de CloudWatch.
- En el editor visual de trabajos, en la pestaña Ejecuciones para un trabajo, elija los hipervínculos para ver los registros:
  - Registros: enlaces a los registros de trabajo de Apache Spark escritos cuando se habilita el registro continuo para una ejecución de trabajo. Este enlace lo dirige a los registros de Amazon CloudWatch en el grupo de registros `/aws-glue/jobs/logs-v2`. De forma predeterminada, los registros excluyen los mensajes de registro de latido de Apache Hadoop YARN no útiles y de ejecutor o controlador de Apache Spark. Para obtener más información acerca del registro continuo, consulte Registro continuo para trabajos de AWS Glue en la Guía para desarrolladores de AWS Glue.
  - Registros de errores: enlaza con los registros escritos en `stderr` para esta ejecución de trabajo. Este enlace lo dirige a los registros de Amazon CloudWatch en el grupo de registros `/aws-glue/jobs/error`. Puede utilizar estos registros para ver detalles acerca de los errores que se encontraron durante la ejecución del trabajo.
  - Registros de salida: enlaza con los registros escritos en `stdout` para esta ejecución de trabajo. Este enlace lo dirige a los registros de Amazon CloudWatch en el grupo de registros `/aws-glue/jobs/output`. Puede utilizar esto registros para ver todos los detalles acerca de las tablas que se crearon en el AWS Glue Data Catalog y los errores que se encontraron.

Visualización de los detalles de una ejecución de trabajo

Puede elegir un trabajo de la lista Ejecuciones de trabajo en la página Monitoreo y, a continuación, elegir Ver detalles de la ejecución para ver información detallada sobre esa ejecución del trabajo.

La información que se muestra en la página detalles de ejecución de trabajo incluye lo siguiente:

Propiedad	Descripción
Nombre de trabajo	El nombre del trabajo.
Estado de ejecución	El estado actual de la ejecución de flujo de trabajo. Los valores pueden ser: • `STARTING` • `RUNNING` • `STOPPING` • `STOPPED` • `SUCCEEDED` • `FAILED` • `TIMEOUT`
Versión de Glue	La versión de AWS Glue utilizada por la ejecución del trabajo.
Intento reciente	El número de intentos de reintento automático para esta ejecución de trabajo.
Hora de inicio	La fecha y la hora en que se inició la ejecución de este flujo de trabajo.
Hora de finalización	La fecha y la hora en que se completó la ejecución de este trabajo.
Hora de inicio	El tiempo empleado en la preparación para ejecutar el trabajo.

Propiedad	Descripción
Hora de ejecución	El tiempo empleado en la ejecución del script de trabajo.
Nombre del disparador	El nombre del desencadenador asociado con el trabajo.
Hora de la última modificación	La fecha en la que se modificó el trabajo por última vez.
Configuración de seguridad	La configuración de seguridad del trabajo, que incluye configuraciones de cifrado de Amazon S3, cifrado de CloudWatch y cifrado de marcadores de trabajo.
Tiempo de espera	El valor del umbral de tiempo de espera de ejecución del trabajo.
Capacidad asignada	El número de unidades de procesamiento de datos (DPU) de AWS Glue asignadas a esta ejecución de trabajo. Para obtener más información acerca de la planificación de capacidad, consulte Monitoreo para planificar la capacidad de DPU en la Guía para desarroll adores de AWS Glue.
Capacidad máxima	La capacidad máxima disponible para la ejecución del trabajo.
Número de procesos de trabajo	El número de empleados utilizados para la ejecución del trabajo.

Propiedad	Descripción
Tipo de empleado	Tipo de empleados predefinidos asignados para la ejecución del trabajo. Los valores pueden ser `G.1X` o `G.2X`. • **`G.1X`**: al elegir este tipo, también debe proporcionar un valor para Número de empleados. Cada proceso de trabajo se asocia a una DPU (4 vCPU, 16 GB de memoria, disco de 64 GB) y proporciona un ejecutor por proceso de trabajo. Le recomendamos este tipo de proceso de trabajo para trabajos con un uso intensivo de la memoria. Esta es la opción predeterminada de Tipo de empleado para trabajos de AWS Glue versión 2.0 o posterior. • **`G.2X`**: al elegir este tipo, también debe proporcionar un valor para Number of workers (Número de empleados). Cada trabajo se asocia a 2 DPU (8 vCPU, 32 GB de memoria, disco de 128 GB) y proporciona un ejecutor por proceso de trabajo. Recomendamos este tipo de empleado para trabajos con una utilización intensiva de la memoria y trabajos que ejecuten transformaciones de machine learning.
Registros	Un enlace a los registros de trabajo para el registro continuo (`/aws-glue/jobs/logs-v2`).
Registros de salida	Un enlace a los archivos de registro de salida del trabajo (`/aws-glue/jobs/output`).
Registros de errores	Un enlace a los archivos de registro de error del trabajo (`/aws-glue/jobs/error`).

También puede ver los siguientes elementos adicionales, que también están disponibles al consultar la información de las ejecuciones de trabajos recientes. Para obtener más información, consulte the section called “Ver información sobre las ejecuciones de trabajos recientes”.

- Argumentos de entrada
- Registros continuos
- Métricas: puede ver visualizaciones de métricas básicas. Para obtener más información sobre las métricas incluidas, consulte the section called “Visualización de métricas de Amazon CloudWatch para una ejecución de trabajo de Spark”.
- Interfaz de usuario de Spark: puede visualizar los registros de Spark para su trabajo en la interfaz de usuario de Spark. Para obtener más información acerca de cómo usar la interfaz de usuario de Spark Web, consulte the section called “Monitorización con la interfaz de usuario de Spark”. Habilite esta característica; para ello, siga el procedimiento que se describe en the section called “Habilitación de la interfaz de usuario de Spark para trabajos”.

Visualización de métricas de Amazon CloudWatch para una ejecución de trabajo de Spark

En la página de detalles de una ejecución de trabajo, debajo de la sección Detalles de ejecución, puede consultar las métricas de trabajo. AWS Glue Studio las envía a Amazon CloudWatch para cada ejecución de trabajo.

AWS Glue notifica las métricas a Amazon CloudWatch cada 30 segundos. Las métricas de AWS Glue representan valores delta que se obtienen de los valores notificados con anterioridad. Si procede, los paneles de métricas acumulan (suman) los valores de 30 segundos para obtener un valor para el último minuto en su totalidad. Sin embargo, las métricas de Apache Spark que AWS Glue transfiere a Amazon CloudWatch, suelen ser valores absolutos que representan el estado actual en el momento en que se notifican.

> **Note**
>
> Debe configurar su cuenta para acceder a Amazon CloudWatch.

Las métricas proporcionan información sobre la ejecución de trabajo, como:

- Movimiento de datos de ETL: número de bytes leídos o escritos en Amazon S3.

- Perfil de la memoria: montón utilizado: el número de bytes de memoria utilizados por el montón de máquina virtual Java (JVM).
- Perfil de la memoria: uso del montón: la fracción de memoria (escala: 0-1), mostrada como porcentaje, utilizada por el montón de JVM.
- Carga de CPU: la fracción de carga del sistema de CPU usada (escala: 0-1), mostrada como porcentaje.

Visualización de métricas de Amazon CloudWatch para una ejecución de trabajo de Ray

En la página de detalles de una ejecución de trabajo, debajo de la sección Detalles de ejecución, puede consultar las métricas de trabajo. AWS Glue Studio las envía a Amazon CloudWatch para cada ejecución de trabajo.

AWS Glue notifica las métricas a Amazon CloudWatch cada 30 segundos. Las métricas de AWS Glue representan valores delta que se obtienen de los valores notificados con anterioridad. Si procede, los paneles de métricas acumulan (suman) los valores de 30 segundos para obtener un valor para el último minuto en su totalidad. Sin embargo, las métricas de Apache Spark que AWS Glue transfiere a Amazon CloudWatch, suelen ser valores absolutos que representan el estado actual en el momento en que se notifican.

> **Note**
>
> Debe configurar su cuenta para acceder a Amazon CloudWatch, tal y como se describe en .

En los trabajos de Ray, puede ver los siguientes gráficos de métricas agregadas. Con ellos, puede crear un perfil del clúster y sus tareas y acceder a información detallada sobre cada nodo. Los datos de serie temporal que respaldan estos gráficos están disponibles en CloudWatch para el análisis posterior.

Perfil de la tarea: estado de la tarea

Muestra el número de tareas de Ray en el sistema. El ciclo de vida de cada tarea tiene su propia serie temporal.

Perfil de la tarea: nombre de la tarea

Muestra el número de tareas de Ray en el sistema. Solo se muestran las tareas pendientes y activas. A cada tipo de tarea (por su nombre) se asigna su propia serie temporal.

Perfil de clúster: CPU en uso

Muestra el número de núcleos de CPU que se utilizan. A cada nodo se asigna su propia serie temporal. Los nodos se identifican mediante direcciones IP, que son efímeras y solo se utilizan para identificación.

Perfil de clúster: uso de memoria del almacén de objetos

Muestra el uso de memoria por parte de la caché de objetos de Ray. A cada ubicación de memoria (memoria física, almacenada en caché en el disco y volcada a Amazon S3) recibe su propia serie temporal. El almacén de objetos administra el almacenamiento de datos en todos los nodos del clúster. Para obtener más información, consulte Objetos en la documentación de Ray.

Perfil de clúster: recuento de nodos

Muestra la cantidad de nodos aprovisionados para el clúster.

Detalle del nodo: uso de la CPU

Muestra el uso de la CPU en cada nodo como porcentaje. Cada serie muestra un porcentaje agregado del uso de la CPU en todos los núcleos del nodo.

Detalle del nodo: uso de memoria

Muestra el uso de memoria (en GB) en cada nodo. Cada serie muestra la memoria agregada entre todos los procesos del nodo, incluidas las tareas de Ray y el proceso de almacenamiento de Plasma. Esto no reflejará los objetos almacenados en el disco o volcados a Amazon S3.

Detalle del nodo: uso del disco

Muestra el uso del disco (en GB) en cada nodo.

Detalle del nodo: velocidad de E/S del disco

Muestra las E/S del disco (en kB/s) en cada nodo.

Detalle del nodo: rendimiento de E/S de red

Muestra las E/S de red (en kB/s) en cada nodo.

Detalle del nodo: uso de la CPU por el componente de Ray

Muestra el uso de la CPU en fracciones de un núcleo. A cada componente de Ray en cada nodo se asigna su propia serie temporal.

Detalle del nodo: uso de la memoria por el componente de Ray

Muestra el uso de memoria en GiB. A cada componente de Ray en cada nodo se asigna su propia serie temporal.

Detectar y procesar información confidencial

La transformación Detectar PII identifica la información de identificación personal (PII) en su origen de datos. Usted elige la entidad PII que desea identificar, cómo desea que se escaneen los datos y qué hacer con la entidad PII identificada por la transformación Detectar PII.

La transformación Detectar PII proporciona la capacidad de detectar, enmascarar o eliminar entidades definidas por el usuario o predefinidas por AWS. Esto permite aumentar la conformidad y reducir la responsabilidad. Por ejemplo, es posible que desee asegurarse de que no exista información de identificación personal en sus datos que pueda leerse y que desee enmascarar los números de seguro social con una cadena fija (como xxx-xx-xxxx), números de teléfono o direcciones.

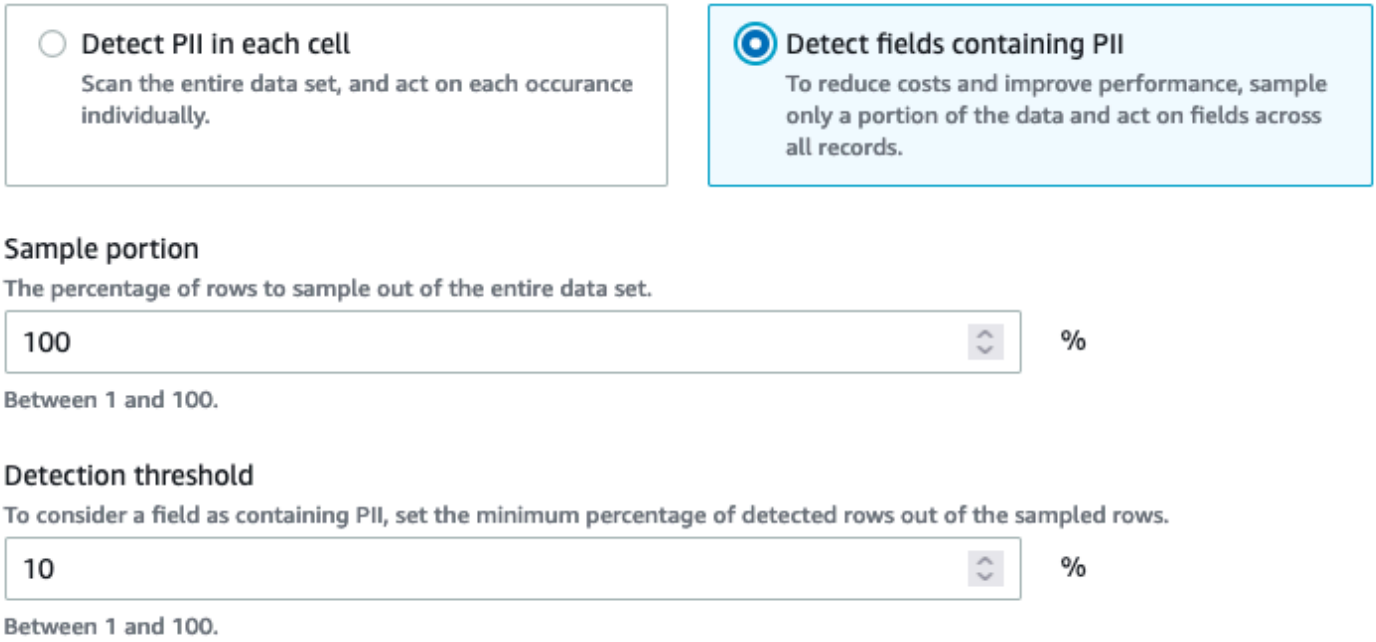
Para trabajar con datos confidenciales fuera de AWS Glue Studio, consulte Uso de la detección de datos confidenciales fuera de AWS Glue Studio

Temas

- Elegir cómo desea que se escaneen los datos
- Elección de las entidades de PII que se desea detectar
- Especificar el nivel de sensibilidad de detección
- Elegir qué hacer con los datos de PII identificados
- Agregar anulaciones de acciones detalladas

Elegir cómo desea que se escaneen los datos

Cuando escanea su conjunto de datos en busca de datos confidenciales, como información de identificación personal (PII), puede elegir detectar la PII en cada fila o detectar las columnas que contienen datos de PII.



Cuando elige Detectar PII en cada celda, elige escanear todas las filas del origen de datos. Se trata de un análisis exhaustivo para garantizar que se identifiquen las entidades de PII.

Cuando elige Detectar campos que contienen PII, elige escanear una muestra de filas en busca de entidades de PII. Esta es una forma de mantener bajos los costos y los recursos, al mismo tiempo que se identifican los campos en los que se encuentran las entidades de PII.

Cuando elige detectar campos que contienen PII, puede reducir los costos y mejorar el rendimiento mediante muestras de una parte de las filas. La elección de esta opción permitirá que especifique opciones adicionales:

- Porción de muestra: permite especificar el porcentaje de filas que se van a mostrar. Por ejemplo, si ingresa ‘50’, especifica que desea el 50 por ciento de las filas escaneadas para la entidad PII.
- Umbrales de detección: permite especificar el porcentaje de filas que contienen la entidad PII para identificar que toda la columna tiene la entidad PII. Por ejemplo, si ingresa ‘10’, especifica que el número de la entidad PII, teléfono EE. UU., en las filas escaneadas debe ser del 10 por ciento o superior para que el campo se identifique como la entidad PII, teléfono EE. UU. Si el porcentaje de filas que contienen la entidad PII es inferior al 10 por ciento, ese campo no se etiquetará por tener la entidad PII, teléfono EE. UU., en él.

Elección de las entidades de PII que se desea detectar

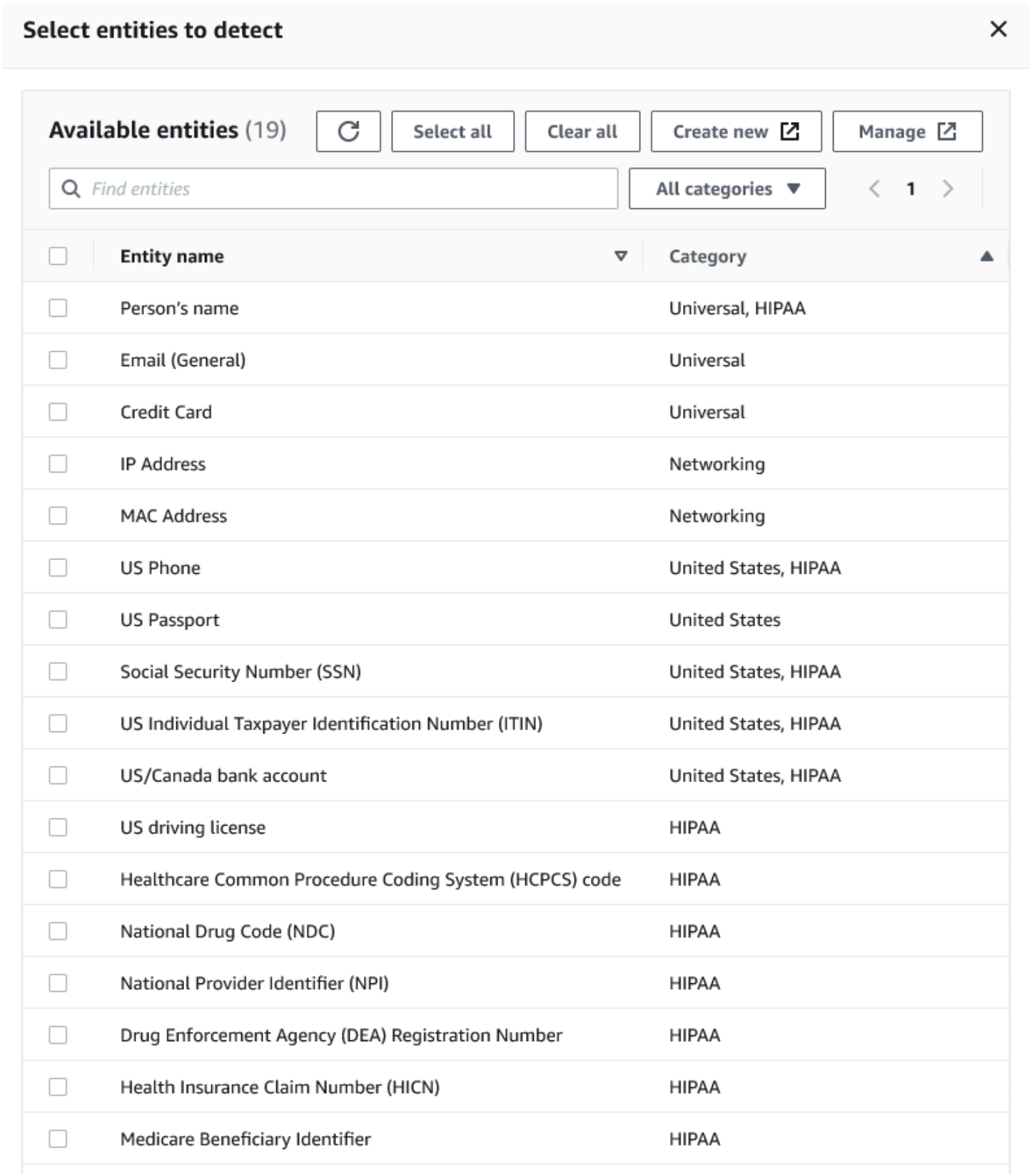
Si ha seleccionado Detectar PII en cada celda, puede elegir entre tres opciones:

- Todos los patrones de PII disponibles: esto incluye a AWS las entidades.
- Seleccionar categorías: si selecciona las categorías, los patrones de PII incluirán automáticamente los patrones de las categorías seleccionadas.
- Seleccionar patrones específicos: solo se detectarán los patrones que seleccione.

Para obtener una lista completa de los tipos de datos confidenciales administrados, consulte Tipos de datos administrados.

Elegir entre todos los patrones de PII disponibles

Si elige Todos los patrones de PII disponibles, seleccione las entidades predefinidas por. AWS Puede seleccionar una entidad, varias o todas ellas.


Select entities to detect
Available entities (19)
Select all
Clear all
Create new
Manage
Find entities
All categories
1
Entity name
Category
Person's name
Universal, HIPAA
Email (General)
Universal
Credit Card
Universal
IP Address
Networking
MAC Address
Networking
US Phone
United States, HIPAA
US Passport
United States
Social Security Number (SSN)
United States, HIPAA
US Individual Taxpayer Identification Number (ITIN)
United States, HIPAA
US/Canada bank account
United States, HIPAA
US driving license
HIPAA
Healthcare Common Procedure Coding System (HCPCS) code
HIPAA
National Drug Code (NDC)
HIPAA
National Provider Identifier (NPI)
HIPAA
Drug Enforcement Agency (DEA) Registration Number
HIPAA
Health Insurance Claim Number (HICN)
HIPAA
Medicare Beneficiary Identifier
HIPAA

Seleccionar categorías

Si ha elegido Seleccionar categorías como patrones de PII que se deben detectar, puede elegir entre las opciones del menú desplegable. Tenga en cuenta que algunas entidades pueden pertenecer a más de una categoría. Por ejemplo, Nombre de la persona es una entidad que pertenece a las categorías Universal y HIPAA.

- Universal (ejemplos: correo electrónico, tarjeta de crédito)
- HIPAA (ejemplos: licencia de conducir de EE. UU., código del sistema de codificación de procedimientos comunes de atención médica [HCPCS])
- Redes (ejemplos: dirección IP, dirección MAC)
- Argentina
- Australia
- Austria
- Bélgica
- Bosnia
- Bulgaria
- Canadá
- Chile
- Colombia
- Croacia
- Chipre
- Chequia
- Dinamarca
- Estonia
- Finlandia
- Francia
- Alemania
- Grecia
- Hungría
- Irlanda

- Corea
- Japón
- México
- Países Bajos
- Nueva Zelanda
- Noruega
- Portugal
- Rumanía
- Singapur
- Eslovaquia
- Eslovenia
- España
- Suecia
- Suiza
- Turquía
- Ucrania
- Estados Unidos
- Reino Unido
- Venezuela

Seleccionar patrones específicos

Si elige Seleccionar patrones específicos como patrones de PII que se deben detectar, puede buscar o examinar una lista de patrones que ya haya creado, o bien crear un nuevo patrón de entidad de detección.

En los siguientes pasos se describe la creación de un nuevo patrón personalizado para detectar información confidencial. Creará el patrón personalizado al ingresar un nombre para dicho patrón, agregará una expresión regular y, opcionalmente, definirá palabras de contexto.

1. Para crear un nuevo patrón, haga clic en el botón Crear nuevo.

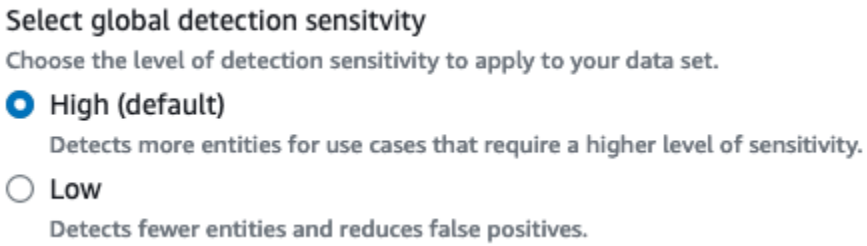
2. En la página Crear entidad de detección, ingrese el nombre de la entidad y una expresión regular. La expresión regular (regex) es lo que AWS Glue utilizará para buscar coincidencias de entidades.
3. Haga clic en Validar. Si la validación se realiza correctamente, aparecerá un mensaje de confirmación que indica que la cadena es una expresión regular válida. Si la validación no se realiza correctamente, aparecerá un mensaje que indica que la cadena no se ajusta al formato adecuado ni a los caracteres, los operadores o las construcciones aceptados.
4. Puede optar por agregar palabras contextuales además de la expresión regular. Las palabras contextuales pueden aumentar la probabilidad de coincidencia. Pueden resultar útiles en los casos en que los nombres de los campos no ofrezcan información descriptiva sobre la entidad. Por ejemplo, los números de la seguridad social pueden denominarse "NSS" o "SS". Agregar estas palabras contextuales puede ayudar a encontrar una coincidencia de la entidad.
5. Haga clic en Crear para crear la entidad de detección. Todas las entidades creadas son visibles en la consola de AWS Glue Studio. Haga clic en Entidades de detección en el menú de navegación de la izquierda.

   Puede editar, eliminar o crear entidades de detección desde la página Entidades de detección. También puede buscar un patrón mediante el campo de búsqueda.

Especificar el nivel de sensibilidad de detección

Puede establecer el nivel de sensibilidad al utilizar la detección de datos confidenciales.

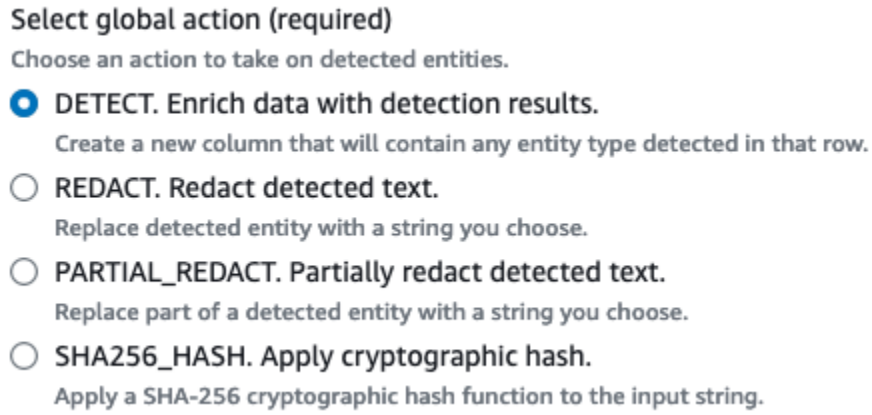
- Alto: (predeterminado) detecta más entidades para los casos de uso que requieren un mayor nivel de sensibilidad. Todos los trabajos de AWS Glue creados después de noviembre de 2023 se activan automáticamente en esta configuración.
- Bajo: detecta menos entidades y reduce los falsos positivos.



Elegir qué hacer con los datos de PII identificados

Si optó por detectar PII en todo el origen de datos, puede elegir una acción global a aplicar:

- Enriquecer los datos con los resultados de detección: si eligió Detectar PII en cada celda, puede almacenar las entidades detectadas en una nueva columna.
- Texto detectado de redacción: puede reemplazar el valor de PII detectado por una cadena que especifique en el campo opcional Reemplazo de entrada de texto. Si no se especifica ninguna cadena, la entidad PII detectada se sustituye por '*******'.
- Texto detectado de redacción: puede reemplazar parte del valor de PII detectado por una cadena que elija. Hay dos opciones posibles: dejar los extremos desenmascarados o enmascararlos proporcionando un patrón de expresiones regulares explícito. Esta característica todavía no está disponible en AWS Glue 2.0.
- Aplicar hash criptográfico: puede pasar el valor de PII detectado a una función de hash criptográfico SHA-256 y reemplazar el valor por la salida de la función.



Diferencias entre AWS Glue las versiones 2.0 y 3.0+

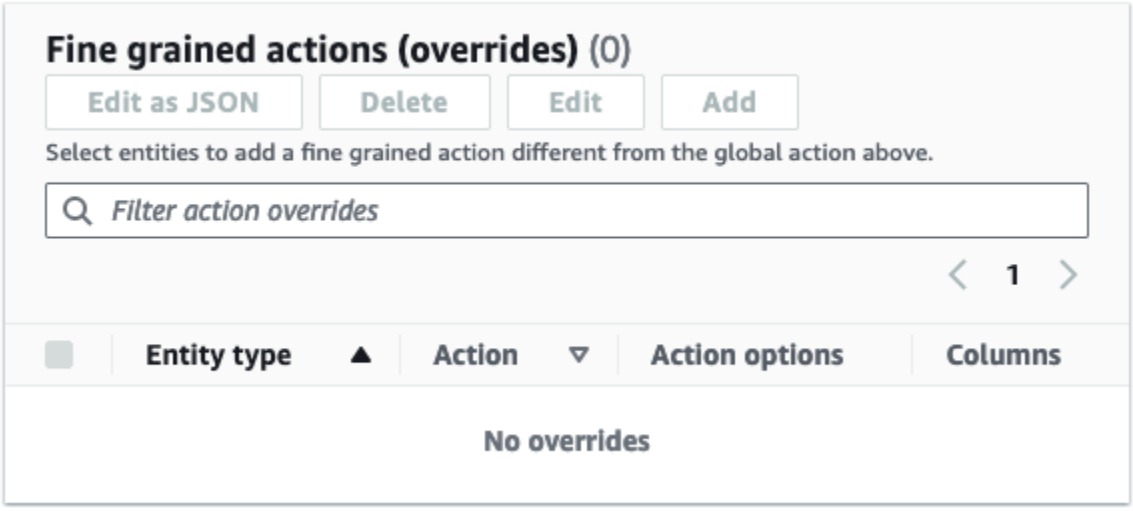
AWS GlueLos trabajos 2.0 devolverán uno nuevo DataFrame con la información de PII detectada para cada columna en una columna complementaria. Cualquier redacción o trabajo hash está visible dentro del guion de AWS Glue en la pestaña visual.

AWS GlueLos trabajos 3.0 y 4.0 devolverán una nueva DataFrame con esta misma columna complementaria. Hay una nueva clave para "actionUsed" y puede ser una de las siguientes: `DETECT`, `REDACT`, `PARTIAL_REDACT` o `SHA256_HASH`. Si se selecciona una acción de enmascaramiento, DataFrame devolverá los datos con los datos confidenciales enmascarados.

Agregar anulaciones de acciones detalladas

Se pueden añadir ajustes adicionales de detección y acción a la tabla de anulaciones de acciones detalladas. Esto le permite:

- Incluir o excluir determinadas columnas de la detección: un esquema inferido en el origen de datos rellenará la tabla con las columnas disponibles.
- Especifique ajustes específicos que sean más detallados que mediante acciones globales. Por ejemplo, puede especificar distintos ajustes de texto de redacción para distintos tipos de entidades.
- Especifique una acción diferente a la acción global: si desea aplicar una acción diferente a un tipo de datos confidenciales diferente, puede hacerlo aquí. Tenga en cuenta que no se pueden usar dos edit-in-place acciones diferentes (redacción y codificación) en la misma columna, pero siempre se puede usar la detección.



Administración de trabajos de ETL con AWS Glue Studio

Puede usar la interfaz gráfica simple en AWS Glue Studio para administrar los trabajos de ETL. En el menú de navegación, seleccione Trabajos para ver la página Trabajos. En esta página, puede ver todos los trabajos que ha creado con AWS Glue Studio o con la consola de AWS Glue. Puede ver, administrar y ejecutar sus trabajos en esta página.

También puede realizar las siguientes acciones:

- Iniciar una ejecución de trabajo
- Programar ejecuciones de trabajo
- Administrar programaciones de trabajo
- Detener ejecuciones de trabajo
- Ver los trabajos
- Ver información sobre las ejecuciones de trabajos recientes
- Ver el script de trabajo
- Modificar las propiedades del trabajo
- Guardar el trabajo
- Clonación de un trabajo
- Eliminación de trabajos

Iniciar una ejecución de trabajo

En AWS Glue Studio, puede ejecutar los trabajos bajo demanda. Un trabajo puede ejecutarse varias veces y cada vez que ejecute el trabajo, AWS Glue recopila información sobre las actividades y el rendimiento del trabajo. Esta información se conoce como una ejecución de trabajo y se identifica mediante un ID de ejecución de trabajo.

Puede iniciar una ejecución de trabajo de las siguientes maneras en AWS Glue Studio:

- En la página Trabajos, elija el trabajo que desea iniciar y, luego, elija el botón Ejecutar trabajo.
- Si está viendo un trabajo en el editor visual y el trabajo se ha guardado, puede elegir el botón Ejecución para iniciar una ejecución de trabajo.

Para obtener más información acerca de las ejecuciones de trabajos, consulte Uso de trabajos en la consola de AWS Glue en la Guía para desarrolladores de AWS Glue.

Programar ejecuciones de trabajo

En AWS Glue Studio, puede crear una programación para que los trabajos se ejecuten en momentos específicos. Puede especificar restricciones, como la cantidad de veces que desea que se ejecute un trabajo, qué días de la semana se ejecutarán y a qué hora. Estas restricciones se basan en `cron` y tienen las mismas limitaciones que `cron`. Por ejemplo, si decide ejecutar su trabajo el día 31 de cada mes, tenga en cuenta que algunos meses no tienen 31 días. Para obtener más información acerca de `cron`, consulte Expresiones Cron en la Guía para desarrolladores de AWS Glue.

Para ejecutar trabajos de acuerdo con una programación

1. Utilice uno de los métodos siguientes para crear una programación de trabajo:
   - En la página Trabajos elija el trabajo para el que desea crear una programación, elija Acciones y, a continuación, elija Programar trabajo.
   - Si está viendo un trabajo en el editor visual y el trabajo se ha guardado, elija la pestaña Programaciones. A continuación, elija Crear programación.
2. En la página Programar ejecución de trabajo, escriba la siguiente información:
   - Nombre: ingrese un nombre para su programación de trabajo.
   - Frecuencia: ingrese la frecuencia para la programación de trabajo. Puede elegir entre las siguientes opciones:
     - Por hora: el trabajo se ejecutará cada hora, comenzando en un minuto específico. Puede especificar el Minuto de la hora que debe ejecutarse el trabajo. De forma predeterminada, cuando elige por hora, el trabajo se ejecuta al comenzar la hora (minuto 0).
     - Por día: el trabajo se ejecutará todos los días, comenzando a la hora indicada. Puede especificar el Minuto de la que hora en la que debe ejecutarse el trabajo y la Hora de inicio para el trabajo. Las horas se especifican con un reloj de 23 horas, en el que se utilizan los números de 13 a 23 para las horas de la tarde. Los valores predeterminados para minuto y hora son 0, lo que significa que si selecciona Por día, el trabajo se ejecutará a medianoche de manera predeterminada.
     - Por semana: el trabajo se ejecutará cada semana en uno o más días. Además de la misma configuración descrita anteriormente para Por día, puede elegir los días de la semana en los que se ejecutará el trabajo. Puede elegir uno o más días.
     - Por mes: el trabajo se ejecutará todos los meses en un día específico. Además de la misma configuración descrita anteriormente para Por día, puede elegir el día del mes en el que se ejecutará el trabajo. Especifique el día como un valor numérico del 1 al 31. Si selecciona

un día que no existe en un mes, por ejemplo, el 30 de febrero, entonces el trabajo no se ejecuta ese mes.

- Personalizado: ingrese una expresión para la programación del trabajo mediante la sintaxis de `cron`. Las expresiones Cron le permiten crear programaciones más complejas, como el último día del mes (en lugar de un día específico del mes) o cada tres meses los días 7 y 21 del mes.

  Consulte Expresiones Cron en la Guía para desarrolladores de AWS Glue

- Descripción: de forma opcional, puede especificar una descripción para la programación de trabajos. Si planea utilizar la misma programación para múltiples trabajos, incluir una descripción facilita determinar las tareas de una programación del trabajo.

3. Elija Crear programación para guardar la programación de trabajos.
4. Después de crear la programación, aparece un mensaje de éxito en la parte superior de la página de la consola. Puede elegir Detalles del trabajo en este banner para ver los detalles del trabajo. Esto abre la página del editor visual de trabajos, con la pestaña Programaciones seleccionada.

Administrar programaciones de trabajo

Después de crear programaciones para un trabajo, puede abrir el trabajo en el editor visual y elegir la pestaña Programaciones para administrar las programaciones.

En la pestaña Programaciones en el editor visual, puede llevar a cabo las siguientes tareas:

- Crear una nueva programación.

  Elija Crear programación y, a continuación, ingrese la información de su programación tal y como se describe en the section called “Programar ejecuciones de trabajo”.

- Edición de una programación existente.

  Elija la programación que desea editar y, a continuación, elija Acción y luego Editar programación. Cuando elija editar una programación existente, Frecuencia aparece como Personalizado, y la programación se muestra como una expresión `cron`. Puede modificar la expresión `cron`, o especificar una nueva programación mediante el botón Frecuencia. Cuando termine de realizar los cambios, elija Actualizar programación.

- Pausar una programación activa.

Elija una programación activa y, a continuación, elija Acción, y luego Pausar programación. La programación se desactiva en forma instantánea. Elija el botón actualizar (recargar) para ver el estado actualizado de la programación de trabajos.

- Reanudar una programación en pausa.

  Elija una programación desactivada y, a continuación, elija Acción, y luego Reanudar programación. La programación se activa en forma instantánea. Elija el botón actualizar (recargar) para ver el estado actualizado de la programación de trabajos.

- Eliminar una programación.

  Elija la programación que desea eliminar y, a continuación, elija Acción y luego Eliminar programación. La programación se elimina en forma instantánea. Elija el botón actualizar (recargar) para ver la programación de trabajo actualizada. La programación mostrará un estado de Eliminación hasta que se haya eliminado por completo.

Detener ejecuciones de trabajo

Puede detener un trabajo antes de que haya completado su ejecución. Puede elegir esta opción si sabe que el trabajo no está configurado correctamente o si el trabajo tarda demasiado en completarse.

En la página Monitoreo, en la lista Ejecuciones de trabajo, elija el trabajo que desea detener, elija Acciones y, a continuación, elija Detener ejecución.

Ver los trabajos

Puede ver todos sus trabajos en la página Trabajos. Puede acceder a esta página al seleccionar Trabajos en el panel de navegación.

En la página Trabajos, puede ver todos los trabajos que se crearon en su cuenta. La lista Sus trabajos muestra el nombre del trabajo, su tipo, el estado de la última ejecución de ese trabajo y las fechas en las que se creó y modificó por última vez el trabajo. Puede elegir el nombre de un trabajo para ver información detallada de ese trabajo.

También puede utilizar el panel Monitoreo para ver todos los trabajos. Puede acceder al panel al elegir Monitoreo en el panel de navegación.

Personalizar la visualización del trabajo

Puede personalizar la forma en que se muestran los trabajos en la sección Sus trabajos en la página Trabajos. Además, puede escribir texto en el campo de texto de búsqueda para mostrar sólo los trabajos con un nombre que contenga ese texto.

Si elige el ícono de configuración

en la sección Sus trabajos, puede personalizar cómo AWS Glue Studio muestra la información en la tabla. Puede elegir ajustar las líneas de texto en la pantalla, cambiar el número de trabajos mostrados en la página y especificar qué columnas mostrar.

Ver información sobre las ejecuciones de trabajos recientes

Un trabajo puede ejecutarse varias veces a medida que se agregan nuevos datos en la ubicación de origen. Cada vez que se ejecuta un trabajo, se le asigna un ID único y se recopila información sobre esa ejecución. Puede utilizar los siguientes métodos para ver esta información:

- Elija la pestaña Ejecuciones del editor visual para ver la información de ejecución del trabajo que se muestra actualmente.

  En la pestaña Ejecuciones (página Ejecuciones de trabajo recientes), se incluye una tarjeta para cada trabajo ejecutado. La información que se muestra en la pestaña Ejecuciones incluye lo siguiente:
  - ID de ejecución del trabajo
  - La cantidad de intentos de ejecución de este trabajo
  - Estado de la ejecución del trabajo
  - Hora de inicio y finalización de la ejecución del trabajo
  - El tiempo de ejecución para la ejecución del trabajo
  - Un enlace a los archivos de registro del trabajo
  - Un enlace a los archivos de registro de error del trabajo
  - Error devuelto por trabajos fallidos
- Puede seleccionar una ejecución del trabajo para ver información adicional, que incluya lo siguiente:
  - Argumentos de entrada
  - Registros continuos

- Métricas: puede ver visualizaciones de métricas básicas. Para obtener más información sobre las métricas incluidas, consulte the section called “Visualización de métricas de Amazon CloudWatch para una ejecución de trabajo de Spark”.
- Interfaz de usuario de Spark: puede visualizar los registros de Spark para su trabajo en la interfaz de usuario de Spark. Para obtener más información acerca de cómo usar la interfaz de usuario de Spark Web, consulte the section called “Monitorización con la interfaz de usuario de Spark”. Habilite esta característica; para ello, siga el procedimiento que se describe en the section called “Habilitación de la interfaz de usuario de Spark para trabajos”.

Puede seleccionar Ver detalles para ver información similar en la página de detalles de la ejecución del trabajo. Como alternativa, puede ir a la página de detalles de la ejecución del trabajo a través de la página Supervisión. En el panel de navegación, seleccione Monitoreo. Desplácese hacia abajo en la lista Ejecuciones de trabajo. Elija el trabajo y, a continuación, elija Ver detalles de ejecución. Los contenidos se describen en Visualización de los detalles de una ejecución de trabajo.

Para obtener más información acerca de los registros de trabajo, consulte Visualización de los registros de ejecución de trabajo.

Ver el script de trabajo

Después de proporcionar información para todos los nodos del trabajo, AWS Glue Studio genera un script que el trabajo utiliza para leer los datos de la fuente, transformarlos y escribirlos en la ubicación de destino. Si guarda el trabajo, puede ver este script en cualquier momento.

Para ver el script generado para su trabajo

1. En el panel de navegación, elija Trabajos.
2. En la página Trabajos, en la lista Sus trabajos elija el nombre del trabajo que desea revisar. Como alternativa, puede seleccionar un trabajo en la lista, elegir la opción Acciones y, a continuación, elegir Editar trabajo.
3. En la página del editor visual, elija la pestaña Script en la parte superior para ver el script de trabajo.

   Si desea editar el script de trabajo, consulte Guía de programación de AWS Glue.

Modificar las propiedades del trabajo

Los nodos del diagrama de trabajo definen las acciones que realiza el trabajo, pero también se pueden configurar varias propiedades para el trabajo. Estas propiedades determinan el entorno en el que se ejecuta el trabajo, los recursos que utiliza, la configuración de umbral, la configuración de seguridad, etc.

Para personalizar el entorno de ejecución del trabajo

1. En el panel de navegación, elija Jobs (Trabajos).
2. En la página Jobs (Trabajos), en la lista Your jobs (Sus trabajos) elija el nombre del trabajo que desea revisar.
3. En la página del editor visual, elija la pestaña Detalles del trabajo en la parte superior del panel de edición del trabajo.
4. Modifique las propiedades del trabajo, según sea necesario.

   Para obtener más información acerca de las propiedades de trabajo, consulte Definición de las propiedades del trabajo en la Guía para desarrolladores de AWS Glue.
5. Expanda la sección Propiedades avanzadas si necesita especificar estas propiedades adicionales del trabajo:

   - Nombre del archivo de script: el nombre del archivo que almacena el script de trabajo en Amazon S3.
   - Ruta del script: ubicación de Amazon S3 donde se almacena el script de trabajo.
   - Métricas de trabajo: (no disponible para trabajos de intérprete de comandos de Python) activa la creación de métricas de Amazon CloudWatch cuando se ejecuta este trabajo.
   - Registro continuo: (no disponible para trabajos de intérprete de comandos de Python) activa el registro continuo en CloudWatch, para que los registros estén disponibles para su visualización antes de que finalice el trabajo.
   - Interfaz de usuario de Spark y Ruta de registros de la interfaz de usuario de Spark: (no disponible para trabajos de intérprete de comandos de Python) activa el uso de la interfaz de usuario de Spark para supervisar este trabajo y especifica la ubicación de los registros de la interfaz de usuario de Spark.
   - Concurrencia máxima: establece el número máximo de ejecuciones concurrentes que están permitidas para este trabajo.

- Ruta temporal: la ubicación de un directorio de trabajo en Amazon S3 donde los resultados intermedios temporales se escriben cuando AWS Glue ejecuta el script de trabajo.
- Umbral de notificación de retraso (minutos): especifica un umbral de retraso para el trabajo. Si el trabajo se ejecuta durante un tiempo más largo que el especificado por el umbral, entonces, AWS Glue envía una notificación de retraso para el trabajo a CloudWatch.
- Configuración de seguridad y Cifrado en el lado del servidor: utilice estos campos para elegir las opciones de cifrado para el trabajo.
- Utilizar Glue Data Catalog como metaalmacén de Hive: elija esta opción si desea utilizar AWS Glue Data Catalog como una alternativa a Apache Hive Metastore.
- Conexión de red adicional: para un origen de datos en una VPC, puede especificar una conexión de tipo `Network`, a fin de garantizar que su trabajo acceda a sus datos a través de la VPC.
- Ruta de la biblioteca Python, Ruta de archivos JAR dependientes (no disponible para trabajos de intérprete de comandos de Python), o Ruta de archivos referenciados: utilice estos campos para especificar la ubicación de los archivos adicionales que el trabajo utiliza cuando ejecuta el script.
- Parámetros del trabajo: puede agregar un conjunto de pares de clave-valor que se transfieren como parámetros con denominación al script de trabajo. En las llamadas de Python a AWS Glue API, es mejor transferir los parámetros explícitamente por nombre. Para obtener más información sobre el uso de parámetros en un script de trabajo, consulte Transferencia y acceso a los parámetros de Python en AWS Glue en la Guía para desarrolladores de AWS Glue.
- Etiquetas: puede agregar etiquetas al trabajo para que le resulte más fácil organizarlos e identificarlos.

6. Después de modificar las propiedades del trabajo, guarde el trabajo.

Almacenar archivos de mezclas aleatorias de Spark en Amazon S3

Algunos trabajos de ETL requieren leer y combinar información de diversas particiones, por ejemplo, cuando se utiliza una transformación de combinación. Esta operación se conoce como mezclado aleatorio. Durante una mezcla aleatoria, los datos se escriben en el disco y se transfieren a través de la red. Con AWS Glue, versión 3.0, puede configurar Amazon S3 como ubicación de almacenamiento para estos archivos. AWS Glue proporciona un administrador de mezclas aleatorias que escribe y lee archivos de mezcla aleatoria desde y hacia Amazon S3. La escritura y la lectura de archivos de mezcla aleatoria de Amazon S3 es más lenta (entre un 5 % y un 20 %) en comparación con el disco

local (o Amazon EBS, que está muy optimizado para Amazon EC2). No obstante, Amazon S3 ofrece capacidad de almacenamiento ilimitada, por lo que no tiene que preocuparse por errores de "No `space left on device`" al ejecutar su trabajo.

Para configurar su trabajo de modo que utilice Amazon S3 para archivos de mezcla aleatoria

1. En la página Trabajos, en la lista Sus trabajos elija el nombre del trabajo que desea modificar.
2. En la página del editor visual, elija la pestaña Job details (Detalles del trabajo) en la parte superior del panel de edición del trabajo.

   Desplácese hasta la sección Parámetros del trabajo.
3. Especifique los siguientes pares clave-valor.

   - `--write-shuffle-files-to-s3` — `true`

     Este es el parámetro principal que configura el administrador de mezclas aleatorias en AWS Glue para utilizar los buckets de Amazon S3 para escribir y leer datos aleatorios. Este parámetro tiene un valor predeterminado de `false`.
   - (Optional) `--write-shuffle-spills-to-s3`: `true`

     Este parámetro le permite descargar archivos de desbordamiento en buckets de Amazon S3, lo que proporciona resistencia adicional a su trabajo de Spark en AWS Glue. Esto solo es necesario para cargas de trabajo grandes que provocan grandes desbordamientos al disco. Este parámetro tiene un valor predeterminado de `false`.
   - (Optional) `--conf spark.shuffle.glue.s3ShuffleBucket`: `S3://<shuffle-bucket>`

     Este parámetro especifica el bucket de Amazon S3 que se utilizará al escribir los archivos de mezcla aleatoria. Si no establece este parámetro, la ubicación es la carpeta `shuffle-data` en la ubicación especificada para Ruta temporal (`--TempDir`).

     > **Note**
     >
     > Asegúrese de que la ubicación del bucket de mezcla aleatoria esté en la misma Región de AWS en la que se ejecuta el trabajo.
     > Además, el servicio de mezcla aleatoria no limpia los archivos después de que el trabajo termine de ejecutarse, por lo que debe configurar las políticas de ciclo de vida de almacenamiento de Amazon S3 en la ubicación del bucket de mezcla

aleatoria. Para obtener más información, consulte Administración del ciclo de vida de almacenamiento en la Guía del usuario de Amazon S3.

Guardar el trabajo

Se muestra un globo de color rojo: El trabajo no se ha guardado, a la izquierda del botón Guardar hasta que guarde el trabajo.

Para guardar el trabajo

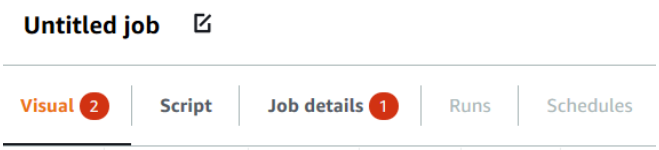
1. Proporcione toda la información necesaria en las pestañas Visual y Detalles del trabajo.
2. Seleccione el botón Guardar.

   Después de guardar el trabajo, el globo “no guardado” cambia para mostrar la hora y la fecha en que se guardó el trabajo por última vez.

Si sale de AWS Glue Studio antes de guardar el trabajo, la próxima vez que inicie sesión en AWS Glue Studio, aparecerá una notificación. La notificación indica que hay un trabajo sin guardar y pregunta si desea restaurarlo. Si decide restaurar el trabajo, podrá continuar editándolo.

Solución de errores al guardar un trabajo

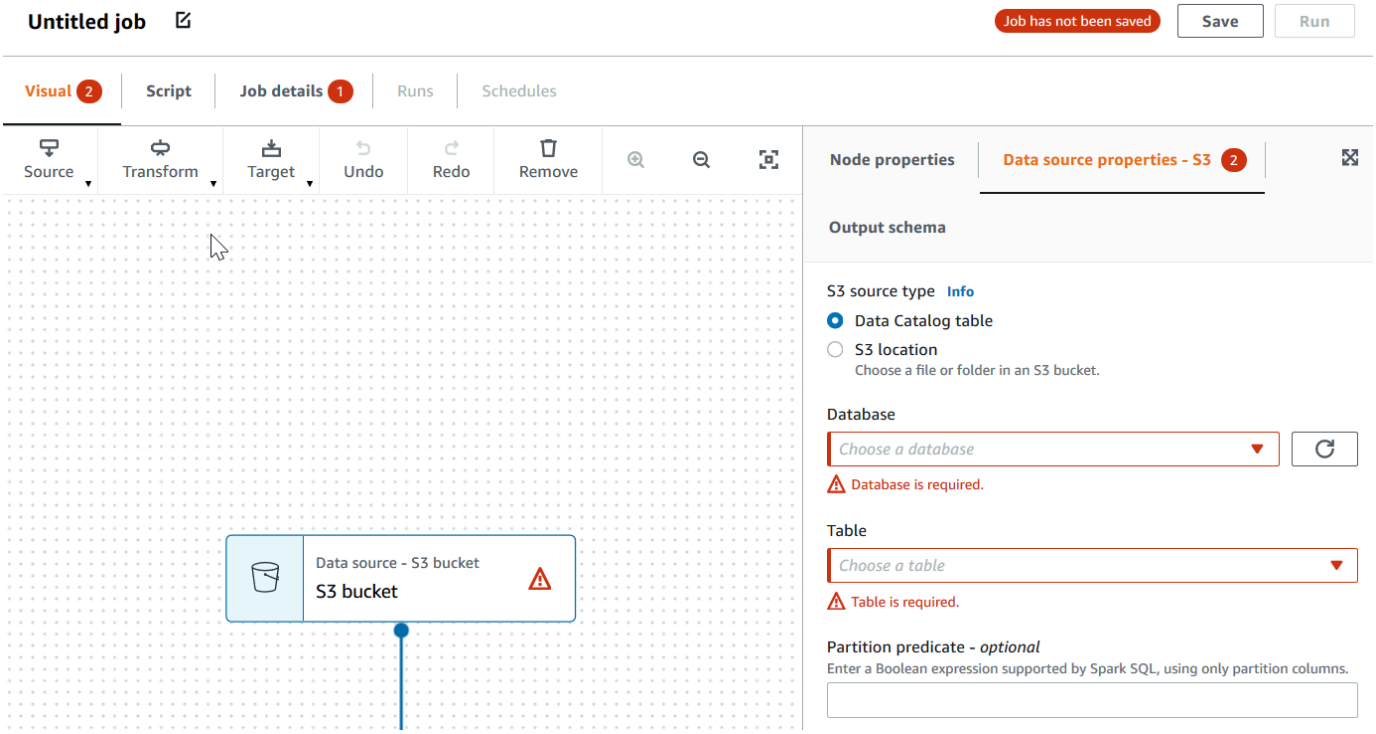
Si elige el botón Guardar, pero a su trabajo le falta información necesaria, aparecerá un globo rojo en la pestaña donde falta la información. El número en el globo indica cuántos campos faltantes se detectaron.



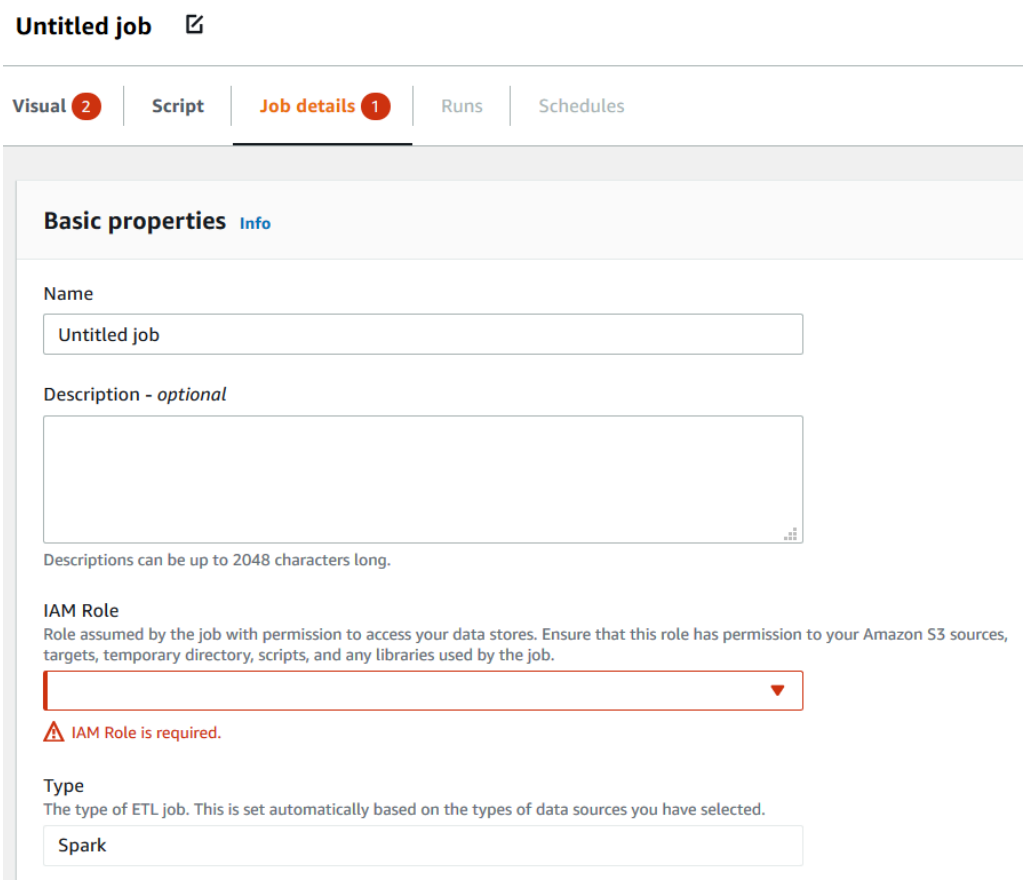
- Si un nodo del editor visual no está configurado correctamente, la pestaña Visual muestra un globo rojo y el nodo con el error muestra un símbolo de advertencia ⚠.

  1. Elija el nodo. En el panel de detalles del nodo, aparece un globo rojo en la pestaña donde se encuentra la información faltante o incorrecta.

2. Elija la pestaña del panel de detalles del nodo que muestra un globo rojo y, a continuación, busque los campos problemáticos, que están resaltados. Un mensaje de error debajo de los campos proporciona información adicional sobre el problema.



- Si hay un problema con las propiedades del trabajo, la pestaña Detalles del trabajo muestra un globo rojo. Elija esa pestaña y busque los campos problemáticos, que están resaltados. Los mensajes de error debajo de los campos proporcionan información adicional acerca del problema.



Clonación de un trabajo

Puede utilizar la acción Clonar trabajo para copiar un trabajo existente en un nuevo trabajo.

Para crear un nuevo trabajo al copiar un trabajo existente

1. En la página Trabajos, en la lista Sus trabajos elija el trabajo que desea duplicar.
2. En el menú Acciones, seleccione Clonar trabajo.
3. Ingrese un nombre para el nuevo trabajo. A continuación, puede guardar o editar el trabajo.

Eliminación de trabajos

Puede eliminar trabajos que ya no son necesarios. Puede eliminar uno o más trabajos en una sola operación.

Para eliminar trabajos de AWS Glue Studio

1. En la página Trabajos, en la lista Sus trabajos elija el trabajo que desea eliminar.
2. En el menú Acciones, elija Eliminar trabajo.
3. Verifique que desea eliminar el trabajo, ingrese **`delete`**.

También puede eliminar un trabajo guardado cuando esté viendo la pestaña Detalles del trabajo de ese trabajo en el editor visual.

Trabajar con tareas en AWS Glue

En las siguientes secciones, se ofrece información acerca de los trabajos de ETL y Ray en AWS Glue.

Temas

- Versiones de AWS Glue
- Trabajar con Spark jobs en AWS Glue
- Trabajar con tareas de Ray en AWS Glue
- Configuración de las propiedades de trabajos del intérprete de comandos de Python en AWS Glue
- Supervisión de AWS Glue
- Estados de ejecución de trabajos de AWS Glue

Versiones de AWS Glue

Puede configurar el parámetro de versión de AWS Glue al agregar o actualizar un trabajo. La versión de AWS Glue determina las versiones de Apache Spark y Python que admite AWS Glue. La versión de Python indica la versión admitida para trabajos de tipo Spark. En la siguiente tabla se muestran las versiones de AWS Glue disponibles, las versiones de Spark y Python correspondientes y otros cambios en la funcionalidad.

Versiones de AWS Glue

AWS Glue versión	Versiones de entorno de tiempo de ejecución compatibles	Versión de Java compatible	Cambios en la funcionalidad
AWS Glue4.0	Versiones del entorno Spark • Spark 3.3.0 • Python 3.10	Java 8	AWS Glue 4.0 es la versión más reciente de AWS Glue. Hay varias optimizaciones y actualizaciones integradas en esta versión de AWS Glue, como las siguientes:

AWS Glue versión	Versiones de entorno de tiempo de ejecución compatibles	Versión de Java compatible	Cambios en la funcionalidad

			• Muchas actualizaciones de funcionalidad de Spark, de Spark 3.1 a Spark 3.3:
			  • Varias mejoras de funcionalidad cuando se combina con Pandas. Para obtener más información, consulte What's New in Spark 3.3 (Novedades de Spark 3.3).
			  • Optimizaciones adicionales desarrolladas en Amazon EMR.
			  • Actualización del sistema de archivos EMR (EMRFS) 2.53.
			• Migración a Log4j 2 desde Log4j 1.x
			• Varias actualizaciones de módulos de Python desde AWS Glue 3.0, como una versión

AWS Glue versión	Versiones de entorno de tiempo de ejecución compatibles	Versión de Java compatible	Cambios en la funcionalidad
			actualizada de Boto. • Actualización de varios conectores, incluido el conector predeterminado de Amazon Redshift. Consulte Apéndice C: actualizaciones del conector. • Actualización de varios controlad ores JDBC. Consulte Apéndice B: actualizaciones de controladores JDBC. • Actualización con un nuevo conector de Amazon Redshift y un controlador JDBC. • Compatibilidad nativa con marcos de lagos de datos de código abierto con Apache Hudi, Delta Lake y Apache Iceberg. • Compatibilidad nativa con el

AWS Glue versión	Versiones de entorno de tiempo de ejecución compatibles	Versión de Java compatible	Cambios en la funcionalidad
			complemento Cloud Shuffle Storage basado en Amazon S3 (un complemento de Apache Spark) a fin de usar Amazon S3 para la mezcla aleatoria y la capacidad de almacenamiento elástica. Limitaciones A continuación, se indican las limitaciones con AWS Glue 4.0: • Las transformaciones de información de identificación personal (PII) y machine learning de AWS Glue aún no están disponibles en AWS Glue 4.0. Para obtener más información sobre cómo migrar a la

AWS Glue versión	Versiones de entorno de tiempo de ejecución compatibles	Versión de Java compatible	Cambios en la funcionalidad
			versión 4.0 de AWS Glue, consulte Migración de trabajos de AWS Glue para Spark a la versión 4.0 de AWS Glue.

AWS Glue versión	Versiones de entorno de tiempo de ejecución compatibles	Versión de Java compatible	Cambios en la funcionalidad
	Versiones del entorno Ray • Ray 2.4.0 Python 3.9	N/A	Cree y ejecute aplicaciones Python distribuidas con AWS Glue for Ray. • Soporta la distribución de datos Ray-2.4.0 (`ray[data]`) con Python 3.9. Para obtener más información sobre esta versión de Ray, consulte Ray-2.4.0 en el repositorio de Ray. GitHub • Admite la instalación de bibliotecas Python adicionales en el entorno de tiempo de ejecución `Ray2.4`. Para obtener más información, consulte the section called “Módulos de Python adicionales para trabajos de Ray”. • Integra registros y métricas de

AWS Glue versión	Versiones de entorno de tiempo de ejecución compatibles	Versión de Java compatible	Cambios en la funcionalidad
			Ray Jobs con Amazon CloudWatch. Para obtener más información, consulte the section called “Solución de errores de Ray” y the section called “Métricas de trabajo de Ray”. • Agrega y visualiza las métricas de los trabajos de Ray en cada AWS Glue Studio página de ejecución de trabajos. • Admite la distribución de archivos en cada directorio de trabajo del clúster, el volcado de objetos del almacén de objetos de Ray a Amazon S3 y el control de la cantidad mínima de nodos de trabajo asignados al trabajo de Ray. Para obtener más información,

AWS Glue versión	Versiones de entorno de tiempo de ejecución compatibles	Versión de Java compatible	Cambios en la funcionalidad
			consulte the section called “Parámetros de los trabajos de Ray”. Limitaciones de los trabajos de Ray en AWS Glue 4.0 • AWS Glue Las sesiones interactivas de Ray permanecen en la versión preliminar de esta versión. • AWS Glue la integración de for Ray con Amazon VPC no está disponible actualmente. No se AWS podrá acceder a los recursos de una VPC sin una ruta pública. Para obtener más información sobre el uso AWS Glue con Amazon VPC, consulte. the section called “Puntos de

AWS Glue versión	Versiones de entorno de tiempo de ejecución compatibles	Versión de Java compatible	Cambios en la funcionalidad
			conexión de VPC (AWS PrivateLink)” • AWS Glue for Ray está disponible en EE.UU. Este (Norte de Virginia), EE.UU. Este (Ohio), EE.UU. Oeste (Oregón), Asia-Pacífico (Tokio) y Europa (Irlanda).

AWS Glue versión	Versiones de entorno de tiempo de ejecución compatibles	Versión de Java compatible	Cambios en la funcionalidad
AWS Glue3.0	• Spark 3.1.1 • Python 3.7	Java 8	Además de la actualización del motor Spark a 3.0, esta versión de AWS Glue ofrece optimizaciones y actualizaciones integradas, como por ejemplo: • Construye la biblioteca de ETL de AWS Glue respecto de Spark 3.0, que es una mejora importante para Spark. • AWS Glue 3.0 soporta los trabajos de streaming. • Incluye nuevas optimizaciones de tiempo de ejecución de AWS Glue Spark para rendimiento y fiabilidad: • Procesamiento columnar en memoria más rápido basado en Apache Arrow para leer datos CSV.

AWS Glue versión	Versiones de entorno de tiempo de ejecución compatibles	Versión de Java compatible	Cambios en la funcionalidad
			• Ejecución basada en SIMD para lecturas vectorizadas con datos CSV. • La actualización de Spark también incluye optimizaciones adicionales desarrolladas en Amazon EMR. • EMRFS actualizado de 2.38 a 2.46, lo que permite nuevas funciones y correcciones de errores para el acceso a Amazon S3. • Se actualizaron varias dependencias necesarias para la nueva versión de Spark. Consulte Apéndice A: actualizaciones de dependencias importantes. • Controladores JDBC actualizados para nuestros

AWS Glue versión	Versiones de entorno de tiempo de ejecución compatibles	Versión de Java compatible	Cambios en la funcionalidad
			orígenes de datos soportados de forma nativa. Consulte Apéndice B: actualizaciones de controladores JDBC. Limitaciones A continuación se indican las limitaciones con AWS Glue 3.0: • Las transformaciones de machine learning de AWS Glue aún no están disponibles en AWS Glue 3.0. • Algunos conectores de Spark personalizados no funcionan con AWS Glue 3.0 si dependen de Spark 2.4 y no son compatibles con Spark 3.1. Para obtener más información sobre cómo migrar a AWS

AWS Glue versión	Versiones de entorno de tiempo de ejecución compatibles	Versión de Java compatible	Cambios en la funcionalidad
			Glue versión 3.0, consulte Migración de trabajos de AWS Glue para Spark a la versión 3.0 de AWS Glue.

AWS Glue versión	Versiones de entorno de tiempo de ejecución compatibles	Versión de Java compatible	Cambios en la funcionalidad
AWS Glue 2.0 (obsoleto, fin del soporte)	• Spark 2.4.3 • Python 3.7	N/A	Además de las características proporcionadas en la versión 1.0 de AWS Glue, la versión 2.0 de AWS Glue también ofrece lo siguiente: • Una infraestructura actualizada para ejecutar trabajos de ETL de Apache Spark en AWS Glue con tiempos de inicio reducidos. • El registro predeterminado es ahora en tiempo real, con flujos separados para controladores y ejecutores, y salidas y errores. • Soporte para especificar módulos de Python adicionales o diferentes versiones al nivel de trabajo.

AWS Glue versión	Versiones de entorno de tiempo de ejecución compatibles	Versión de Java compatible	Cambios en la funcionalidad
			Note La versión 2.0 de AWS Glue difiere de la versión 1.0 de AWS Glue para algunas dependencias y versiones debido a cambios arquitectónicos subyacentes. Valide los trabajos de AWS Glue antes de migrar a versiones de AWS Glue posteriores. Para obtener más información acerca de las características y limitaciones de la versión 2.0 de AWS Glue, consulte Ejecución de trabajos de ETL de Spark

AWS Glue versión	Versiones de entorno de tiempo de ejecución compatibles	Versión de Java compatible	Cambios en la funcionalidad
			con tiempos de inicio reducidos.

AWS Glue versión	Versiones de entorno de tiempo de ejecución compatibles	Versión de Java compatible	Cambios en la funcionalidad
AWS Glue 1.0 (obsoleto, fin del soporte)	• Spark 2.4.3 • Python 2.7 • Python 3.6	N/A	Puede mantener marcadores de trabajo para formatos Parquet y ORC en trabajos de ETL de AWS Glue (con AWS Glue versión 1.0). Anteriormente, solo podía marcar formatos de origen de Amazon S3 comunes, como JSON, CSV, Apache Avro y XML en trabajos de ETL de AWS Glue. Si configura opciones de formato para las entradas y salidas de ETL, puede especific ar que se utilice el formato del lector/es critor de Apache Avro 1.8 para poder leer y escribir tipos lógicos de Avro (con AWS Glue versión 1.0). Anteriormente, solo se admitía el formato del lector/escritor de la versión 1.7 de Avro.

AWS Glue versión	Versiones de entorno de tiempo de ejecución compatibles	Versión de Java compatible	Cambios en la funcionalidad
			El tipo de conexión de DynamoDB admite una opción de escritura (con la versión 1.0 de AWS Glue). Limitaciones A continuación, se indican las limitaciones con AWS Glue 1.0: • Las versiones 0.9 y 1.0 de AWS Glue no estarán disponibles en Asia Pacífico (Yakarta) (`ap-southeast-3`), Oriente Medio (EAU) (`me-central-1`) ni en otras regiones nuevas en el futuro.

AWS Glue versión	Versiones de entorno de tiempo de ejecución compatibles	Versión de Java compatible	Cambios en la funcionalidad
AWS Glue 0.9 (obsoleto, fin del soporte)	• Spark 2.2.1 • Python 2.7	N/A	Los trabajos que se crearon sin especificar una versión de AWS Glue se han establecido de forma predeterminada en AWS Glue 0.9. Limitaciones A continuación, se indican las limitaciones con AWS Glue 0.9: • Las versiones 0.9 y 1.0 de AWS Glue no estarán disponibles en Asia Pacífico (Yakarta) (`ap-southeast-3`), Oriente Medio (EAU) (`me-central-1`) ni en otras regiones nuevas en el futuro.

Ejecución de trabajos de ETL de Spark con tiempos de inicio reducidos

La versión 2.0 de AWS Glue y posteriores proporcionan una infraestructura mejorada para ejecutar los trabajos de Apache Spark del servicio ETL (extracción, transformación y carga) en AWS Glue con tiempos de inicio reducidos. Con los tiempos de espera reducidos, los ingenieros de datos pueden ser más productivos y aumentar su interactividad con AWS Glue. La reducción de la variación en los

tiempos de inicio de los trabajos puede ayudarlo a cumplir o superar los acuerdos de nivel de servicio respecto de la disponibilidad de datos para análisis.

Para utilizar esta característica con los trabajos de ETL de AWS Glue, elija **`2.0`** o una versión posterior para la `Glue version` cuando cree sus trabajos.

Temas

- Características nuevas admitidas
- Comportamiento de registro
- Características no admitidas

Características nuevas admitidas

En esta sección, se describen las nuevas características compatibles con las versiones 2.0 de AWS Glue y posteriores.

Compatibilidad con especificación de módulos de Python adicionales al nivel de trabajo

La versión 2.0 de AWS Glue y posteriores también permiten proporcionar módulos de Python adicionales o versiones diferentes al nivel de trabajo. Puede utilizar la opción `--additional-python-modules` con una lista de módulos Python separados por comas para agregar un nuevo módulo o cambiar la versión de un módulo existente.

Por ejemplo, para actualizar o agregar un nuevo módulo `scikit-learn` utilice la siguiente clave/valor: `"--additional-python-modules", "scikit-learn==0.21.3"`.

Además, dentro de la opción `--additional-python-modules`, puede especificar una ruta de Amazon S3 a un módulo de wheel de Python. Por ejemplo:

```
--additional-python-modules s3://aws-glue-native-spark/tests/j4.2/ephem-3.7.7.1-cp37-cp37m-linux_x86_64.whl,s3://aws-glue-native-spark/tests/j4.2/fbprophet-0.6-py3-none-any.whl,scikit-learn==0.21.3
```

AWS Glue utiliza el instalador de paquetes de Python (pip3) para instalar los módulos adicionales. Puede transferir opciones adicionales especificadas por `python-modules-installer-option` a pip3 para instalar los módulos. Será de aplicación cualquier incompatibilidad o limitación de pip3.

Módulos de Python ya proporcionados en AWS Glue versión 2.0

AWS Glue, versión 2.0 soporta los siguientes módulos de Python listos para su uso:

- setuptools—45.2.0
- subprocess32—3.5.4
- ptvsd—4.3.2
- pydevd—1.9.0
- PyMySQL—0.9.3
- docutils—0.15.2
- jmespath—0.9.4
- six—1.14.0
- python_dateutil—2.8.1
- urllib3—1.25.8
- botocore—1.15.4
- s3transfer—0.3.3
- boto3—1.12.4
- certifi—2019.11.28
- chardet—3.0.4
- idna—2.9
- requests—2.23.0
- pyparsing—2.4.6
- enum34—1.1.9
- pytz—2019.3
- numpy—1.18.1
- cycler—0.10.0
- kiwisolver—1.1.0
- scipy—1.4.1
- pandas—1.0.1
- pyarrow—0.16.0
- matplotlib—3.1.3
- pyhocon—0.3.54

- mpmath—1.1.0
- sympy—1.5.1
- patsy—0.5.1
- statsmodels—0.11.1
- fsspec—0.6.2
- s3f—0.4.0
- Cython—0.29.15
- joblib—0.14.1
- pmdarima—1.5.3
- scikit-learn—0.22.1
- tbat—1.0.9

Comportamiento de registro

La versión 2.0 de AWS Glue y posteriores admiten diferentes comportamientos de registro predeterminados. Las diferencias incluyen:

- El registro se produce en tiempo real.
- Hay flujos separados para controladores y ejecutores.
- Para cada controlador y ejecutor hay dos flujos, el flujo de salida y el flujo de error.

Flujos de controladores y ejecutores

El ID de ejecución del trabajo identifica los flujos de controladores. Los flujos del ejecutor se identifican mediante el <*run id*>_<*executor task id*> del trabajo. Por ejemplo:

- `"logStreamName": "jr_8255308b426fff1b4e09e00e0bd5612b1b4ec848d7884cebe61ed33a31789..._g-f65f617bd31d54bd94482af755b6cdf464542..."`

Flujos de salida y errores

El flujo de salida tiene la salida estándar (stdout) de su código. El flujo de error tiene mensajes de registro de su código/biblioteca.

- Flujos de registro:
  - Los flujos de registro de controladores tienen <*jr*>, donde <*jr*> es el ID de la ejecución de trabajo.
  - Los flujos de registro del ejecutor tienen <*jr*>_<*g*>, donde <*g*> es el ID de tarea del ejecutor. Puede buscar el ID de tarea del ejecutor en el registro de errores del controlador.

Los grupos de registros predeterminados para AWS Glue, versión 2.0, son los siguientes:

- `/aws-glue/jobs/logs/output` para la salida
- `/aws-glue/jobs/logs/error` para los errores

Cuando se proporciona una configuración de seguridad, los nombres de los grupos de registro cambian a:

- `/aws-glue/jobs/<security configuration>-role/<Role Name>/output`
- `/aws-glue/jobs/<security configuration>-role/<Role Name>/error`

En la consola, el enlace Logs (Registros) vincula al grupo de registro de salida y el enlace Error vincula al grupo de registros de errores. Cuando se habilita el registro continuo, el enlace Logs (Registros) vincula al grupo de registros continuos, y el enlace Output (Salida) vincula al grupo de registro de salida.

Reglas de registro

> **Note**
>
> El nombre de grupo de registro predeterminado para el registro continuo es `/aws-glue/jobs/logs-v2`.

En las versiones 2.0 de AWS Glue y posteriores, el registro continuo tiene el mismo comportamiento que en la versión 1.0 de AWS Glue:

- Grupo de registros predeterminado: `/aws-glue/jobs/logs-v2`
- Flujo de registro del controlador: <*jr*>-driver
- Flujo de registro del ejecutor: <*jr*>-<*executor ID*>

Se puede cambiar el nombre del grupo de registros si configura `--continuous-log-logGroupName`

El nombre de los flujos de registro puede tener un prefijo si configura `--continous-log-logStreamPrefix`

Características no admitidas

Las siguientes características de AWS Glue no son compatibles:

- Puntos de enlace de desarrollo
- Las versiones 2.0 de AWS Glue y posteriores no se ejecutan en Apache YARN, por lo que la configuración de YARN no se aplica
- Las versiones 2.0 de AWS Glue y posteriores no tienen Hadoop Distributed File System (HDFS)
- Las versiones 2.0 de AWS Glue y posteriores no utilizan asignación dinámica, por lo tanto, las métricas ExecutorAllocationManager no están disponibles
- Para trabajos de versiones 2.0 de AWS Glue y posteriores, se debe especificar el número de empleados y el tipo de empleado, pero no la `maxCapacity`.
- Las versiones 2.0 de AWS Glue y posteriores no son compatibles con `s3n` sin configurar. Se recomienda utilizar `s3` o `s3a`. Si los trabajos requieren el uso de `s3n` por cualquier motivo, puede transferir el siguiente argumento adicional:

```
--conf spark.hadoop.fs.s3n.impl=com.amazon.ws.emr.hadoop.fs.EmrFileSystem
```

Migración de trabajos de AWS Glue para Spark a la versión 3.0 de AWS Glue

Este tema describe los cambios entre las versiones 0.9, 1.0, 2.0 y 3.0 de AWS Glue para permitirle migrar sus aplicaciones de Spark y trabajos de ETL a AWS Glue 3.0.

Para usar esta función con sus trabajos de ETL de AWS Glue, elija **`3.0`** para la `Glue version` cuando cree sus trabajos.

Temas

- Características nuevas admitidas

- Acciones para migrar a AWS Glue 3.0
- Lista de comprobación de migración
- Migrar de AWS Glue 0.9 a AWS Glue 3.0
- Migrar de AWS Glue 1.0 a AWS Glue 3.0
- Migrar de AWS Glue 2.0 a AWS Glue 3.0
- Apéndice A: actualizaciones de dependencias importantes
- Apéndice B: actualizaciones de controladores JDBC

Características nuevas admitidas

En esta sección se describen las nuevas características y ventajas de AWS Glue, versión 3.0.

- Se basa en Apache Spark 3.1.1, que cuenta con optimizaciones de Spark de código abierto, desarrollado por los servicios de AWS Glue y EMR, tales como ejecución de consultas adaptativas, lectores vectorizados y combinación de particiones y mezclas aleatorias optimizadas.
- Controladores JDBC actualizados para todos los orígenes nativos de Glue, que incluyen MySQL, Microsoft SQL Server, Oracle, PostgreSQL, MongoDB, y bibliotecas y dependencias de Spark actualizadas introducidas por Spark 3.1.1.
- Acceso optimizado a Amazon S3 con EMRFS mejorado y confirmadores de salida optimizados de Amazon S3 de forma predeterminada.
- Acceso optimizado al Data Catalog con índices de partición, predicados de inserción, lista de particiones y cliente de metaalmacén de Hive actualizado.
- Integración con Lake Formation para tablas de catálogo gobernadas con filtrado a nivel de celda y transacciones de lago de datos.
- Experiencia de interfaz de usuario de Spark mejorada con Spark 3.1.1 con nuevas métricas de memoria del ejecutor de Spark y métricas de streaming estructuradas de Spark.
- Reducción de la latencia de inicio que mejora los tiempos generales de finalización del trabajo y la interactividad, similar a AWS Glue 2.0.
- Los trabajos de Spark se facturan en incrementos de 1 segundo con una duración mínima de facturación 10 veces menor, de un mínimo de 10 minutos a un mínimo de 1 minuto, similar a AWS Glue 2.0.

Acciones para migrar a AWS Glue 3.0

Para los trabajos existentes, cambie la `Glue version` de la versión anterior a `Glue 3.0` en la configuración del trabajo.

- En la consola, elija `Spark 3.1, Python 3 (Glue Version 3.0) or Spark 3.1, Scala 2 (Glue Version 3.0)` en `Glue version`.
- En AWS Glue Studio, elija `Glue 3.0 - Supports spark 3.1, Scala 2, Python 3` en `Glue version`.
- En la API, elija **`3.0`** en el parámetro `GlueVersion` en la API de `UpdateJob`.

Para nuevos trabajos, elija `Glue 3.0` cuando cree un trabajo.

- En la consola, elija `Spark 3.1, Python 3 (Glue Version 3.0) or Spark 3.1, Scala 2 (Glue Version 3.0)` en `Glue version`.
- En AWS Glue Studio, elija `Glue 3.0 - Supports spark 3.1, Scala 2, Python 3` en `Glue version`.
- En la API, elija **`3.0`** en el parámetro `GlueVersion` en la API de `CreateJob`.

Para ver los registros de eventos de Spark de AWS Glue 3.0, lanzar un servidor de historial de Spark actualizado para Glue 3.0 mediante CloudFormation o Docker.

Lista de comprobación de migración

Revise esta lista de comprobación para la migración.

- ¿Su trabajo depende de HDFS? En caso afirmativo, intente reemplazar HDFS por S3.
  - Buscar en la ruta del sistema de archivos comenzando con `hdfs://` o `/` como ruta DFS en el código del script del trabajo.
  - Compruebe si su sistema de archivos predeterminado no está configurado con HDFS. Si está explícitamente configurado, debe eliminar la configuración `fs.defaultFS`.
  - Compruebe si su trabajo contiene parámetros `dfs.*`. Si contiene alguno, debe verificar que es correcto desactivar los parámetros.
- ¿Su trabajo depende de YARN? En caso afirmativo, verifique los impactos al comprobar si su trabajo contiene los siguientes parámetros. Si contiene alguno, debe verificar que es correcto desactivar los parámetros.

- `spark.yarn.*`

  Por ejemplo:

  ```
  spark.yarn.executor.memoryOverhead
  spark.yarn.driver.memoryOverhead
  spark.yarn.scheduler.reporterThread.maxFailures
  ```

- `yarn.*`

  Por ejemplo:

  ```
  yarn.scheduler.maximum-allocation-mb
  yarn.nodemanager.resource.memory-mb
  ```

- ¿Su trabajo depende de Spark 2.2.1 o Spark 2.4.3? En caso afirmativo, verifique los impactos al comprobar si su trabajo utiliza características que se modificaron en Spark 3.1.1.
  - `https://spark.apache.org/docs/latest/sql-migration-guide.html#upgrading-from-spark-sql-22-to-23`

    Por ejemplo: la función `percentile_approx`, o `SparkSession` con `SparkSession.builder.getOrCreate()` cuando hay un `SparkContext` existente.
  - `https://spark.apache.org/docs/latest/sql-migration-guide.html#upgrading-from-spark-sql-23-to-24`

    Por ejemplo: la función `array_contains`, o `CURRENT_DATE`, función `CURRENT_TIMESTAMP` con `spark.sql.caseSensitive=true`.
- ¿Los archivos JAR adicionales de su trabajo entran en conflicto en Glue 3.0?
  - A partir de AWS Glue 0.9/1.0: los archivos JAR adicionales suministrados en trabajos existentes de AWS Glue 0.9/1.0 pueden provocar conflictos de classpath debido a dependencias actualizadas o nuevas disponibles en Glue 3.0. Puede evitar conflictos de classpath en AWS Glue 3.0 con el parámetro de trabajo `--user-jars-first` de AWS Glue o al sombrear sus dependencias.
  - A partir de AWS Glue 2.0: puede evitar conflictos de classpath en AWS Glue 3.0 con el parámetro de trabajo `--user-jars-first` de AWS Glue o al sombrear sus dependencias.
- ¿Dependen sus trabajo de Scala 2.11?
  - AWS Glue 3.0 usa Scala 2.12, por lo que necesita reconstruir sus bibliotecas con Scala 2.12, si las mismas dependen de Scala 2.11.

- ¿Las bibliotecas externas de Python de su trabajo dependen de Python 2.7/3.6?
  - Use el parámetro `--additional-python-modules` en lugar de establecer el archivo egg/wheel/zip en la ruta de la biblioteca de Python.
  - Actualice las bibliotecas dependientes de Python 2.7/3.6 a Python 3.7 ya que Spark 3.1.1 eliminó el soporte de Python 2.7.

Migrar de AWS Glue 0.9 a AWS Glue 3.0

Tenga en cuenta los siguientes cambios al migrar:

- AWS Glue 0.9 utiliza Spark 2.2.1 y AWS Glue 3.0 utiliza Spark 3.1.1 optimizado para EMR.
  - Varios cambios de Spark por sí solos pueden requerir la revisión de sus scripts para garantizar que no se haga referencia a las características eliminadas.
  - Por ejemplo, Spark 3.1.1 no habilita UDF sin tipo Scala, pero Spark 2.2 sí los permite.
- Todos los trabajos en AWS Glue 3.0 se ejecutarán con tiempos de inicio significativamente mejorados. Los trabajos de Spark se facturarán en incrementos de 1 segundo con una duración mínima de facturación 10 veces menor, ya que la latencia de inicio pasará de 10 minutos máximo a 1 minuto máximo.
- El comportamiento de registro se ha modificado desde AWS Glue 2.0.
- Varias actualizaciones de dependencias, destacadas en Apéndice A: actualizaciones de dependencias importantes.
- Scala también se actualiza a 2.12 desde 2.11, y Scala 2.12 no es compatible con Scala 2.11.
- Python 3.7 es también la versión predeterminada utilizada para los scripts de Python, ya que AWS Glue 0.9 solo utilizaba Python 2.
  - Spark 3.1.1. no soporta Python 2.7.
  - Se encuentra disponible un nuevo mecanismo para instalar módulos adicionales de Python.
- AWS Glue 3.0 no se ejecuta en Apache YARN, por lo que la configuración de YARN no se aplica.
- AWS Glue 3.0 no tiene un sistema de archivos distribuido Hadoop Distributed File System (HDFS).
- Cualquier archivo JAR adicional suministrado en trabajos de AWS Glue 0.9 puede traer dependencias conflictivas ya que hubo actualizaciones en varias dependencias en 3.0 con respecto a 0.9. Puede evitar conflictos de classpath en AWS Glue 3.0 con el parámetro de trabajo `--user-jars-first` de AWS Glue.
- AWS Glue 3.0 no utiliza asignación dinámica, por lo tanto, las métricas ExecutorAllocationManager no están disponibles.

- Para trabajos de AWS Glue, versión 3.0, debe especificar el número de empleados y el tipo de empleado, pero no la `maxCapacity`.
- Los trabajos de la versión 3.0 de AWS Glue no soportan transformaciones de machine learning.
- AWS Glue 3.0 aún no soporta puntos de enlace de desarrollo.

Consulte la documentación de migración de Spark:

- consulte Actualización de Spark SQL 2.2 a 2.3
- consulte Actualización de Spark SQL 2.3 a 2.4
- consulte Actualización de Spark SQL 2.4 a 3.0
- consulte Actualización de Spark SQL 3.0 a 3.1
- consulte Cambios en el comportamiento Datetime que se esperan a partir de Spark 3.0.

Migrar de AWS Glue 1.0 a AWS Glue 3.0

Tenga en cuenta los siguientes cambios al migrar:

- AWS Glue 1.0 utiliza Spark 2.4 de código abierto y AWS Glue 3.0 utiliza Spark 3.1.1 optimizado para EMR.
  - Varios cambios de Spark por sí solos pueden requerir la revisión de sus scripts para garantizar que no se haga referencia a las características eliminadas.
  - Por ejemplo, Spark 3.1.1 no habilita UDF sin tipo Scala, pero Spark 2.4 sí los permite.
- Todos los trabajos en AWS Glue 3.0 se ejecutarán con tiempos de inicio significativamente mejorados. Los trabajos de Spark se facturarán en incrementos de 1 segundo con una duración mínima de facturación 10 veces menor, ya que la latencia de inicio pasará de 10 minutos máximo a 1 minuto máximo.
- El comportamiento de registro se ha modificado desde AWS Glue 2.0.
- Varias actualizaciones de dependencias, destacadas en
- Scala también se actualiza a 2.12 desde 2.11, y Scala 2.12 no es compatible con Scala 2.11.
- Python 3.7 es también la versión predeterminada utilizada para los scripts de Python, ya que AWS Glue 0.9 solo utilizaba Python 2.
  - Spark 3.1.1. no soporta Python 2.7.
  - Se encuentra disponible un nuevo mecanismo para instalar módulos adicionales de Python.
- AWS Glue 3.0 no se ejecuta en Apache YARN, por lo que la configuración de YARN no se aplica.

- AWS Glue 3.0 no tiene un sistema de archivos distribuido Hadoop Distributed File System (HDFS).
- Cualquier archivo JAR adicional suministrado en trabajos de AWS Glue 1.0 puede traer dependencias conflictivas ya que hubo actualizaciones en varias dependencias en 3.0 con respecto a 1.0. Puede evitar conflictos de classpath en AWS Glue 3.0 con el parámetro de trabajo `--user-jars-first` de AWS Glue.
- AWS Glue 3.0 no utiliza asignación dinámica, por lo tanto, las métricas ExecutorAllocationManager no están disponibles.
- Para trabajos de AWS Glue, versión 3.0, debe especificar el número de empleados y el tipo de empleado, pero no la `maxCapacity`.
- Los trabajos de la versión 3.0 de AWS Glue no soportan transformaciones de machine learning.
- AWS Glue 3.0 aún no soporta puntos de enlace de desarrollo.

Consulte la documentación de migración de Spark:

- consulte Actualización de Spark SQL 2.4 a 3.0
- consulte Cambios en el comportamiento Datetime que se esperan a partir de Spark 3.0.

Migrar de AWS Glue 2.0 a AWS Glue 3.0

Tenga en cuenta los siguientes cambios al migrar:

- Todos los parámetros de trabajo existentes y las principales características que existen en AWS Glue 2.0 existirá en AWS Glue 3.0.
  - El compromiso optimizado para S3 de EMRFS para la escritura de datos de Parquet en Amazon S3 está habilitado de forma predeterminada en AWS Glue 3.0. Sin embargo, todavía puede desactivarlo mediante la configuración de `--enable-s3-parquet-optimized-committer` a `false`.
- AWS Glue 2.0 utiliza Spark 2.4 de código abierto y AWS Glue 3.0 utiliza Spark 3.1.1 optimizado para EMR.
  - Varios cambios de Spark por sí solos pueden requerir la revisión de sus scripts para garantizar que no se haga referencia a las características eliminadas.
  - Por ejemplo, Spark 3.1.1 no habilita UDF sin tipo Scala, pero Spark 2.4 sí los permite.
- AWS Glue 3.0 también incluye una actualización de EMRFS, controladores JDBC actualizados e inclusiones de optimizaciones adicionales en Spark proporcionadas por AWS Glue.

- Todos los trabajos en AWS Glue 3.0 se ejecutarán con tiempos de inicio significativamente mejorados. Los trabajos de Spark se facturarán en incrementos de 1 segundo con una duración mínima de facturación 10 veces menor, ya que la latencia de inicio pasará de 10 minutos máximo a 1 minuto máximo.
- Spark 3.1.1. no soporta Python 2.7.
- Varias actualizaciones de dependencias, destacadas en Apéndice A: actualizaciones de dependencias importantes.
- Scala también se actualiza a 2.12 desde 2.11, y Scala 2.12 no es compatible con Scala 2.11.
- Cualquier archivo JAR adicional suministrado en trabajos de AWS Glue 2.0 puede traer dependencias conflictivas ya que hubo actualizaciones en varias dependencias en 3.0 con respecto a 2.0. Puede evitar conflictos de classpath en AWS Glue 3.0 con el parámetro de trabajo `--user-jars-first` de AWS Glue.
- AWS Glue 3.0 tiene un paralelismo de tareas de Spark diferente para la configuración del controlador/ejecutor en comparación con AWS Glue 2.0. También mejora el rendimiento y utiliza mejor los recursos disponibles. Tanto `spark.driver.cores` como `spark.executor.cores` están configurados para el número de núcleos en AWS Glue 3.0 (4 en el estándar y en el empleado `G.1X`, y 8 en el empleado `G.2X`). Estas configuraciones no cambian el tipo de empleado ni el hardware del trabajo de AWS Glue. Puede utilizar estas configuraciones para calcular el número de particiones o divisiones que coinciden con el paralelismo de tareas de Spark en la aplicación Spark.

  En general, los trabajos tendrán un rendimiento similar o mejorado en comparación con la versión 2.0 de AWS Glue. Si los trabajos se ejecutan más lentamente, puede aumentar el paralelismo entre las tareas a través del siguiente argumento de trabajo:

  - clave: `--executor-cores` valor: <*cantidad deseada de tareas que se pueden ejecutar en paralelo*>
  - El valor no debe superar el doble de la cantidad de vCPU en el tipo de trabajador, es decir, 8 en `G.1X`, 16 en `G.2X`, 32 en `G.4X` y 64 en `G.8X`. Debe tener cuidado al actualizar esta configuración, ya que podría afectar el rendimiento laboral porque el aumento del paralelismo entre las tareas provoca presión en la memoria y el disco y podría limitar los sistemas de origen y destino.
- AWS Glue 3.0 utiliza Spark 3.1, que cambia el comportamiento de cargar/guardar marcas de tiempo desde archivos de parquet y hacia ellos. Para obtener más información, consulte Actualización de Spark SQL 3.0 a 3.1

Recomendamos establecer los siguientes parámetros al leer y escribir datos de parquet que contengan columnas de marca temporal. La configuración de estos parámetros puede resolver el problema de incompatibilidad del calendario que se produce durante la actualización de Spark 2 a Spark 3, tanto para el marco dinámico de AWS Glue como para el marco de datos de Spark. Utilice la opción CORRECTED para leer el valor de fecha y hora tal como está; y la opción LEGACY para rebasar los valores de fecha y hora con respecto a la diferencia de calendario durante la lectura.

```
- Key: --conf
- Value: spark.sql.legacy.parquet.int96RebaseModeInRead=[CORRECTED|LEGACY] --
conf spark.sql.legacy.parquet.int96RebaseModeInWrite=[CORRECTED|LEGACY] --conf
 spark.sql.legacy.parquet.datetimeRebaseModeInRead=[CORRECTED|LEGACY]
```

Consulte la documentación de migración de Spark:

- consulte Actualización de Spark SQL 2.4 a 3.0
- consulte Cambios en el comportamiento Datetime que se esperan a partir de Spark 3.0.

Apéndice A: actualizaciones de dependencias importantes

Las siguientes son las actualizaciones de dependencias:

Dependencia	Versión en AWS Glue 0.9	Versión en AWS Glue 1.0	Versión en AWS Glue 2.0	Versión en AWS Glue 3.0
Spark	2.2.1	2.4.3	2.4.3	3.1.1-amzn-0
Hadoop	2.7.3-amzn-6	2.8.5-amzn-1	2.8.5-amzn-5	3.2.1-amzn-3
Scala	2.11	2.11	2.11	2.12
Jackson	2.7.x	2.7.x	2.7.x	2.10.x
Hive	1.2	1.2	1.2	2.3.7-amzn-4
EMRFS	2.20.0	2.30.0	2.38.0	2.46.0
Json4s	3.2.x	3.5.x	3.5.x	3.6.6

Dependencia	Versión en AWS Glue 0.9	Versión en AWS Glue 1.0	Versión en AWS Glue 2.0	Versión en AWS Glue 3.0
Arrow	N/A	0.10.0	0.10.0	2.0.0
Cliente de catálogo de AWS Glue	N/A	N/A	1.10.0	3.0.0

Apéndice B: actualizaciones de controladores JDBC

Las siguientes son las actualizaciones de controladores JDBC:

Controlador	Versión del controlador JDBC en las versiones de AWS Glue anteriores	Versión del controlador JDBC en AWS Glue 3.0
MySQL	5.1	8.0.23
Microsoft SQL Server	6.1.0	7.0.0
Oracle Database	11.2	21.1
PostgreSQL	42.1.0	42.2.18
MongoDB	2.0.0	4.0.0

Migración de trabajos de AWS Glue para Spark a la versión 4.0 de AWS Glue

En este tema se describen los cambios entre las versiones 0.9, 1.0, 2.0 y 3.0 de AWS Glue para permitirle migrar sus aplicaciones de Spark y trabajos de ETL a AWS Glue 4.0. También se describen las características de la versión 4.0 de AWS Glue y las ventajas de usarla.

Para usar esta función con sus trabajos de ETL de AWS Glue, elija **`4.0`** para la `Glue version` cuando cree sus trabajos.

Temas

- Características nuevas admitidas
- Acciones que migrar a AWS Glue 4.0
- Lista de comprobación de migración
- Migración de AWS Glue 3.0 a AWS Glue 4.0
- Migración de AWS Glue 2.0 a AWS Glue 4.0
- Migración de AWS Glue 1.0 a AWS Glue 4.0
- Migración de AWS Glue 0.9 a AWS Glue 4.0
- Migración de conectores y controladores JDBC para AWS Glue 4.0
- Apéndice A: actualizaciones de dependencias importantes
- Apéndice B: actualizaciones de controladores JDBC
- Apéndice C: actualizaciones del conector

Características nuevas admitidas

En esta sección se describen las nuevas características y ventajas de la versión 4.0 de AWS Glue.

- Se basa en Apache Spark 3.3.0, pero incluye optimizaciones en AWS Glue y Amazon EMR, tales como ejecuciones de consultas adaptativas, lectores vectorizados y combinación de particiones y mezclas aleatorias optimizadas.
- Controladores JDBC actualizados para todos los orígenes nativos de AWS Glue, que incluyen MySQL, Microsoft SQL Server, Oracle, PostgreSQL, MongoDB, y bibliotecas y dependencias de Spark actualizadas introducidas por Spark 3.3.0.
- Actualización con un nuevo conector de Amazon Redshift y un controlador JDBC.
- Acceso optimizado a Amazon S3 con el sistema de archivos EMR (EMRFS) actualizado y confirmadores de salida optimizados de Amazon S3 habilitados de forma predeterminada.
- Acceso optimizado al Catálogo de datos con índices de partición, predicados de inserción, lista de particiones y un cliente de metaalmacén de Hive actualizado.
- Integración con Lake Formation para tablas de catálogo gobernadas con filtrado a nivel de celda y transacciones de lago de datos.
- Reducción de la latencia de inicio para mejorar los tiempos generales de finalización del trabajo y la interactividad.
- Los trabajos de Spark se facturan en incrementos de 1 segundo con una duración mínima de facturación 10 veces menor, de un mínimo de 10 minutos a un mínimo de 1 minuto.

- Compatibilidad nativa con marcos de lagos de datos de código abierto con Apache Hudi, Delta Lake y Apache Iceberg.
- Compatibilidad nativa con el complemento Cloud Shuffle Storage basado en Amazon S3 (un complemento de Apache Spark) a fin de usar Amazon S3 para la mezcla aleatoria y la capacidad de almacenamiento elástica.

Mejoras principales de Spark 3.1.1 a Spark 3.3.0

Tenga en cuenta las siguientes mejoras:

- Filtrado en tiempo de ejecución de fila (SPARK-32268).
- Mejoras de ANSI (SPARK-38860).
- Mejoras de los mensajes de error (SPARK-38781).
- Compatibilidad de tipos complejos con el lector vectorizado de Parquet (SPARK-34863).
- Compatibilidad de metadatos de archivos ocultos con Spark SQL (SPARK-37273).
- Se proporciona un generador de perfiles para las UDF de Python o Pandas (SPARK-37443).
- Se introduce Trigger.AvailableNow para ejecutar consultas de transmisión como Trigger.Once en varios lotes (SPARK-36533).
- Capacidades de inserción de Datasource V2 más completas (SPARK-38788).
- Migración de log4j 1 a log4j 2 (SPARK-37814).

Otros cambios notables

Tenga en cuenta los siguientes cambios:

- Cambios bruscos
  - Se eliminan las referencias a la compatibilidad con Python 3.6 en los documentos y documentación de Python (SPARK-36977).
  - Se elimina el truco de tupla con nombre al reemplazar el objeto pickle integrado por cloudpickle (SPARK-32079).
  - Se incrementa la versión mínima de Pandas a la 1.0.5 (SPARK-37465).

Acciones que migrar a AWS Glue 4.0

Para los trabajos existentes, cambie la `Glue version` de la versión anterior a `Glue 4.0` en la configuración del trabajo.

- En AWS Glue Studio, elija `Glue 4.0 - Supports Spark 3.3, Scala 2, Python 3` en `Glue version`.
- En la API, elija **`4.0`** en el parámetro `GlueVersion` de la operación de la API `UpdateJob`.

Para nuevos trabajos, elija `Glue 4.0` cuando cree un trabajo.

- En la consola, elija `Spark 3.3, Python 3 (Glue Version 4.0) or Spark 3.3, Scala 2 (Glue Version 3.0)` en `Glue version`.
- En AWS Glue Studio, elija `Glue 4.0 - Supports Spark 3.3, Scala 2, Python 3` en `Glue version`.
- En la API, elija **`4.0`** en el parámetro `GlueVersion` de la operación de la API `CreateJob`.

Para ver los registros de eventos de Spark de AWS Glue 4.0 procedentes de AWS Glue 2.0 o una versión anterior, lance un servidor de historial de Spark actualizado para AWS Glue 4.0 mediante AWS CloudFormation o Docker.

Lista de comprobación de migración

Revise esta lista de comprobación para la migración:

> **Note**
>
> En el caso de los elementos de la lista de comprobación relacionados con la versión 3.0 de AWS Glue, consulte Lista de comprobación de migración.

- ¿Las bibliotecas externas de Python de su trabajo dependen de Python 2.7/3.6?
  - Actualice las bibliotecas dependientes de Python 2.7 o 3.6 a Python 3.10, ya que Spark 3.3.0 eliminó por completo la compatibilidad con Python 2.7 y 3.6.

Migración de AWS Glue 3.0 a AWS Glue 4.0

Tenga en cuenta los siguientes cambios al migrar:

- Todos los parámetros de trabajo existentes y las principales características que existen en AWS Glue 3.0 existirán en AWS Glue 4.0.
- AWS Glue 3.0 utiliza Spark 3.1.1 optimizado para Amazon EMR y AWS Glue 4.0 utiliza Spark 3.3.0 optimizado para Amazon EMR.

  Es posible que varios cambios de Spark por sí solos requieran la revisión de sus scripts para garantizar que no se haga referencia a las características eliminadas.
- AWS Glue 4.0 también incluye una actualización de EMRFS y Hadoop. Para saber la versión específica, consulte Apéndice A: actualizaciones de dependencias importantes.
- La versión de AWS SDK que se proporciona en los trabajos de ETL ahora se ha actualizado de la 1.11 a la 1.12.
- Todos los trabajos de Python utilizarán la versión 3.10 de Python. Anteriormente, Python 3.7 se usaba en AWS Glue 3.0.

  Como resultado, se actualizan algunos pymodules que AWS Glue incluye listos para usar.
- Log4j se ha actualizado a Log4j2.
  - Para obtener información sobre la ruta de migración de Log4j2, consulte la documentación de Log4j.
  - En su lugar, debe cambiar el nombre de cualquier archivo log4j.properties personalizado a un archivo log4j2.properties, con las propiedades de log4j2 adecuadas.
- Para migrar determinados conectores, consulte Migración de conectores y controladores JDBC para AWS Glue 4.0.
- Se actualizó el SDK de cifrado de AWS de 1.x a 2.x. Se ven afectados los trabajos de AWS Glue que utilizan configuraciones de seguridad de AWS Glue y los trabajos que dependen de la dependencia del SDK de cifrado de AWS proporcionada en tiempo de ejecución. Consulte las instrucciones para la migración de trabajos de AWS Glue.

  Puede actualizar de forma segura un trabajo de AWS Glue 2.0 o 3.0 a un trabajo de AWS Glue 4.0 porque AWS Glue 2.0 o 3.0 ya contiene la versión puente del SDK de cifrado de AWS.

Consulte la documentación de migración de Spark:

- Upgrading from Spark SQL 3.1 to 3.2 (Actualización de Spark SQL 3.1 a 3.2)
- Upgrading from Spark SQL 3.2 to 3.3 (Actualización de Spark SQL 3.2 a 3.3)

Migración de AWS Glue 2.0 a AWS Glue 4.0

Tenga en cuenta los siguientes cambios al migrar:

> **Note**
>
> Para conocer los pasos de migración relacionados con AWS Glue 3.0, consulte Migración de AWS Glue 3.0 a AWS Glue 4.0.

- Todos los parámetros de trabajo existentes y las principales características que existen en AWS Glue 2.0 existirán en AWS Glue 4.0.
- El confirmador optimizado para S3 de EMRFS para la escritura de datos de Parquet en Amazon S3 está habilitado de forma predeterminada a partir de AWS Glue 3.0. Sin embargo, todavía puede desactivarlo mediante la configuración de `--enable-s3-parquet-optimized-committer` a `false`.
- AWS Glue 2.0 utiliza la versión 2.4 de Spark de código abierto y AWS Glue 4.0, la versión 3.3.0 optimizada para Amazon EMR.
  - Varios cambios de Spark por sí solos pueden requerir la revisión de sus scripts para garantizar que no se haga referencia a las características eliminadas.
  - Por ejemplo, Spark 3.3.0 no permite las UDF sin tipo Scala, pero Spark 2.4 sí.
- La versión de AWS SDK que se proporciona en los trabajos de ETL ahora se ha actualizado de la 1.11 a la 1.12.
- AWS Glue 4.0 también incluye una actualización de EMRFS, controladores JDBC actualizados e inclusiones de optimizaciones adicionales en Spark proporcionadas por AWS Glue.
- Scala se actualiza a la versión 2.12 desde la 2.11, y Scala 2.12 no es compatible con Scala 2.11.
- Python 3.10 es la versión predeterminada utilizada para los scripts de Python, ya que AWS Glue 2.0 solo utilizaba Python 3.7 y 2.7.
  - Python 2.7 no es compatible con Spark 3.3.0. Cualquier trabajo que solicite Python 2 en la configuración del trabajo producirá un error con IllegalArgumentException.
  - Hay disponible un nuevo mecanismo para instalar módulos adicionales de Python a partir de AWS Glue 2.0.

- Varias actualizaciones de dependencias, destacadas en Apéndice A: actualizaciones de dependencias importantes.
- Cualquier archivo JAR adicional suministrado en trabajos existentes de AWS Glue 2.0 puede traer dependencias conflictivas, ya que hubo actualizaciones en varias dependencias en la versión 4.0 con respecto a la 2.0. Puede evitar conflictos de classpath en AWS Glue 4.0 con el parámetro de trabajo `--user-jars-first` de AWS Glue.
- AWS Glue 4.0 usa Spark 3.3. A partir de Spark 3.1, se produjo un cambio en el comportamiento de cargar/guardar marcas de tiempo desde archivos de parquet y hacia estos. Para obtener más información, consulte Actualización de Spark SQL 3.0 a 3.1

  Recomendamos establecer los siguientes parámetros al leer y escribir datos de parquet que contengan columnas de marca temporal. La configuración de estos parámetros puede resolver el problema de incompatibilidad del calendario que se produce durante la actualización de Spark 2 a Spark 3, tanto para el marco dinámico de AWS Glue como para el marco de datos de Spark. Utilice la opción CORRECTED para leer el valor de fecha y hora tal como está; y la opción LEGACY para rebasar los valores de fecha y hora con respecto a la diferencia de calendario durante la lectura.

  ```
  - Key: --conf
  - Value: spark.sql.legacy.parquet.int96RebaseModeInRead=[CORRECTED|LEGACY] --
  conf spark.sql.legacy.parquet.int96RebaseModeInWrite=[CORRECTED|LEGACY] --conf
   spark.sql.legacy.parquet.datetimeRebaseModeInRead=[CORRECTED|LEGACY]
  ```
- Para migrar determinados conectores, consulte Migración de conectores y controladores JDBC para AWS Glue 4.0.
- Se actualizó el SDK de cifrado de AWS de 1.x a 2.x. Se ven afectados los trabajos de AWS Glue que utilizan configuraciones de seguridad de AWS Glue y los trabajos que dependen de la dependencia del SDK de cifrado de AWS proporcionada en tiempo de ejecución. Consulte estas instrucciones para la migración de trabajos de AWS Glue:
  - Puede actualizar de forma segura un trabajo de AWS Glue 2.0 a un trabajo de AWS Glue 4.0 porque AWS Glue 2.0 ya contiene la versión puente del SDK de cifrado de AWS.

Consulte la documentación de migración de Spark:

- Upgrading from Spark SQL 2.4 to 3.0 (Actualización de Spark SQL 2.4 a 3.0)
- Upgrading from Spark SQL 3.1 to 3.2 (Actualización de Spark SQL 3.1 a 3.2)
- Upgrading from Spark SQL 3.2 to 3.3 (Actualización de Spark SQL 3.2 a 3.3)

- Changes in Datetime behavior to be expected since Spark 3.0 (Cambios en el comportamiento de Datetime que se esperan a partir de Spark 3.0).

Migración de AWS Glue 1.0 a AWS Glue 4.0

Tenga en cuenta los siguientes cambios al migrar:

- AWS Glue 1.0 utiliza la versión 2.4 de Spark de código abierto y AWS Glue 4.0, la versión 3.3.0 optimizada para Amazon EMR.
  - Varios cambios de Spark por sí solos pueden requerir la revisión de sus scripts para garantizar que no se haga referencia a las características eliminadas.
  - Por ejemplo, Spark 3.3.0 no permite las UDF sin tipo Scala, pero Spark 2.4 sí.
- Todos los trabajos de AWS Glue 4.0 se ejecutarán con tiempos de inicio significativamente mejorados. Los trabajos de Spark se facturarán en incrementos de 1 segundo con una duración mínima de facturación 10 veces menor, ya que la latencia de inicio pasará de 10 minutos máximo a 1 minuto máximo.
- El comportamiento del registro cambió significativamente en AWS Glue 4.0; Spark 3.3.0 tiene un requisito mínimo de Log4j2.
- Varias actualizaciones de dependencias, que se destacan en el apéndice.
- Scala también se actualiza a la versión 2.12 desde la 2.11, y Scala 2.12 no es compatible con Scala 2.11.
- Python 3.10 es también la versión predeterminada utilizada para los scripts de Python, ya que AWS Glue 0.9 solo utilizaba Python 2.

  Python 2.7 no es compatible con Spark 3.3.0. Cualquier trabajo que solicite Python 2 en la configuración del trabajo producirá un error con IllegalArgumentException.
- Hay disponible un nuevo mecanismo para instalar módulos adicionales de Python mediante pip a partir de AWS Glue 2.0. Para obtener más información, consulte Instalación de módulos de Python adicionales con pip en AWS Glue 2.0+.
- AWS Glue 4.0 no se ejecuta en Apache YARN, por lo que la configuración de YARN no se aplica.
- AWS Glue 4.0 no tiene un Sistema de archivos distribuido de Hadoop (HDFS).
- Cualquier archivo JAR adicional suministrado en trabajos existentes de AWS Glue 1.0 puede traer dependencias conflictivas, ya que hubo actualizaciones en varias dependencias en la versión 4.0 con respecto a la 1.0. Habilitamos AWS Glue 4.0 con el parámetro de trabajo de AWS Glue `--user-jars-first` de forma predeterminada para evitar este problema.

- AWS Glue 4.0 admite el escalado automático. Por lo tanto, la métrica ExecutorAllocationManager estará disponible cuando se habilite el escalado automático.
- Para trabajos de la versión 4.0 de AWS Glue, debe especificar el número de procesos de trabajo y el tipo de proceso de trabajo, pero no un valor de `maxCapacity`.
- Los trabajos de AWS Glue 4.0 todavía no admiten transformaciones de machine learning.
- Para migrar determinados conectores, consulte Migración de conectores y controladores JDBC para AWS Glue 4.0.
- Se actualizó el SDK de cifrado de AWS de 1.x a 2.x. Se ven afectados los trabajos de AWS Glue que utilizan configuraciones de seguridad de AWS Glue y los trabajos que dependen de la dependencia del SDK de cifrado de AWS proporcionada en tiempo de ejecución. Consulte estas instrucciones para la migración de trabajos de AWS Glue.
  - No puede migrar ningún trabajo de AWS Glue 0.9 o 1.0 a un trabajo de AWS Glue 4.0 directamente. Esto se debe a que, al actualizar directamente a la versión 2.x o una posterior y habilitar todas las características nuevas de inmediato, el SDK de cifrado de AWS no podrá descifrar el texto cifrado en versiones anteriores del SDK de cifrado de AWS.
  - Para llevar a cabo la actualización de forma segura, primero le recomendamos que migre a un trabajo de AWS Glue 2.0 o 3.0 que contenga la versión puente del SDK de cifrado de AWS. Ejecute el trabajo una vez para utilizar la versión puente del SDK de cifrado de AWS.
  - Al finalizar, puede migrar de forma segura el trabajo de AWS Glue 2.0 o 3.0 a AWS Glue 4.0.

Consulte la documentación de migración de Spark:

- Upgrading from Spark SQL 2.4 to 3.0 (Actualización de Spark SQL 2.4 a 3.0)
- Upgrading from Spark SQL 3.0 to 3.1 (Actualización de Spark SQL 3.0 a 3.1)
- Upgrading from Spark SQL 3.1 to 3.2 (Actualización de Spark SQL 3.1 a 3.2)
- Upgrading from Spark SQL 3.2 to 3.3 (Actualización de Spark SQL 3.2 a 3.3)
- Changes in Datetime behavior to be expected since Spark 3.0 (Cambios en el comportamiento de Datetime que se esperan a partir de Spark 3.0).

Migración de AWS Glue 0.9 a AWS Glue 4.0

Tenga en cuenta los siguientes cambios al migrar:

- AWS Glue 0.9 utiliza la versión 2.2.1 de Spark de código abierto y AWS Glue 4.0, la versión 3.3.0 optimizada para Amazon EMR.

  - Es posible que varios cambios de Spark por sí solos requieran la revisión de sus scripts para garantizar que no se haga referencia a las características eliminadas.
  - Por ejemplo, Spark 3.3.0 no permite las UDF sin tipo Scala, pero Spark 2.2 sí.
- Todos los trabajos de AWS Glue 4.0 se ejecutarán con tiempos de inicio significativamente mejorados. Los trabajos de Spark se facturarán en incrementos de 1 segundo con una duración mínima de facturación 10 veces menor, porque la latencia de inicio pasará de un máximo de 10 minutos a un máximo de 1 minuto.
- El comportamiento del registro cambió significativamente a partir de AWS Glue 4.0; Spark 3.3.0 tiene un requisito mínimo de Log4j2, como se menciona aquí (https://spark.apache.org/docs/latest/core-migration-guide.html#upgrading-from-core-32-to-33).
- Varias actualizaciones de dependencias, que se destacan en el apéndice.
- Scala también se actualiza a la versión 2.12 desde la 2.11, y Scala 2.12 no es compatible con Scala 2.11.
- Python 3.10 es también la versión predeterminada utilizada para los scripts de Python, ya que AWS Glue 0.9 solo utilizaba Python 2.
  - Python 2.7 no es compatible con Spark 3.3.0. Cualquier trabajo que solicite Python 2 en la configuración del trabajo producirá un error con IllegalArgumentException.
  - Hay disponible un nuevo mecanismo para instalar módulos adicionales de Python mediante pip.
- AWS Glue 4.0 no se ejecuta en Apache YARN, por lo que la configuración de YARN no se aplica.
- AWS Glue 4.0 no tiene un Sistema de archivos distribuido de Hadoop (HDFS).
- Cualquier archivo JAR adicional suministrado en trabajos existentes de AWS Glue 0.9 puede traer dependencias conflictivas, ya que hubo actualizaciones en varias dependencias en la versión 3.0 con respecto a la 0.9. Puede evitar conflictos de classpath en AWS Glue 3.0 con el parámetro de trabajo `--user-jars-first` de AWS Glue.
- AWS Glue 4.0 admite el escalado automático. Por lo tanto, la métrica ExecutorAllocationManager estará disponible cuando se habilite el escalado automático.
- Para trabajos de la versión 4.0 de AWS Glue, debe especificar el número de procesos de trabajo y el tipo de proceso de trabajo, pero no un valor de `maxCapacity`.
- Los trabajos de AWS Glue 4.0 todavía no admiten transformaciones de machine learning.
- Para migrar determinados conectores, consulte Migración de conectores y controladores JDBC para AWS Glue 4.0.
- Se actualizó el SDK de cifrado de AWS de 1.x a 2.x. Se ven afectados los trabajos de AWS Glue que utilizan configuraciones de seguridad de AWS Glue y los trabajos que dependen de la

dependencia del SDK de cifrado de AWS proporcionada en tiempo de ejecución. Consulte estas instrucciones para la migración de trabajos de AWS Glue.

- No puede migrar ningún trabajo de AWS Glue 0.9 o 1.0 a un trabajo de AWS Glue 4.0 directamente. Esto se debe a que, al actualizar directamente a la versión 2.x o una posterior y habilitar todas las características nuevas de inmediato, el SDK de cifrado de AWS no podrá descifrar el texto cifrado en versiones anteriores del SDK de cifrado de AWS.
- Para llevar a cabo la actualización de forma segura, primero le recomendamos que migre a un trabajo de AWS Glue 2.0 o 3.0 que contenga la versión puente del SDK de cifrado de AWS. Ejecute el trabajo una vez para utilizar la versión puente del SDK de cifrado de AWS.
- Al finalizar, puede migrar de forma segura el trabajo de AWS Glue 2.0 o 3.0 a AWS Glue 4.0.

Consulte la documentación de migración de Spark:

- Upgrading from Spark SQL 2.2 to 2.3 (Actualización de Spark SQL 2.2 a 2.3)
- Upgrading from Spark SQL 2.3 to 2.4 (Actualización de Spark SQL 2.3 a 2.4)
- Upgrading from Spark SQL 2.4 to 3.0 (Actualización de Spark SQL 2.4 a 3.0)
- Upgrading from Spark SQL 3.0 to 3.1 (Actualización de Spark SQL 3.0 a 3.1)
- Upgrading from Spark SQL 3.1 to 3.2 (Actualización de Spark SQL 3.1 a 3.2)
- Upgrading from Spark SQL 3.2 to 3.3 (Actualización de Spark SQL 3.2 a 3.3)
- Changes in Datetime behavior to be expected since Spark 3.0 (Cambios en el comportamiento de Datetime que se esperan a partir de Spark 3.0).

Migración de conectores y controladores JDBC para AWS Glue 4.0

Para ver las versiones de los conectores de lago de datos y JDBC que se actualizaron, consulte:

- Apéndice B: actualizaciones de controladores JDBC
- Apéndice C: actualizaciones del conector

Hudi

- Mejoras de compatibilidad de Spark SQL:
  - A través del comando `Call Procedure`, se agrega compatibilidad para actualizar, degradar, arrancar, limpiar y reparar. La sintaxis `Create/Drop/Show/Refresh Index` es posible en Spark SQL.

  - Se cerró una laguna de rendimiento entre el uso mediante un DataSource de Spark en comparación con Spark SQL. En el pasado, las escrituras en orígenes de datos solían ser más rápidas que en SQL.
  - Todos los generadores de claves integrados implementan operaciones de API específicas de Spark de mayor rendimiento.
  - Se sustituyó la transformación de UDF en la operación masiva `insert` por transformaciones de RDD para reducir los costos de uso de SerDe.
  - Spark SQL con Hudi requiere que se especifique un campo `primaryKey` mediante `tblproperites` u opciones en la instrucción SQL. En el caso de las operaciones de actualización y eliminación, también se requiere el campo `preCombineField`.
- Cualquier tabla de Hudi creada antes de la versión 0.10.0 sin un campo `primaryKey` tiene que volver a crearse con un campo `primaryKey` a partir de la versión 0.10.0.

PostgreSQL

- Se corrigieron varias vulnerabilidades (CVE).
- Java 8 es compatible de forma nativa.
- Si el trabajo utiliza matrices de matrices, con la excepción de las matrices de bytes, dicho escenario se puede tratar como matrices multidimensionales.

MongoDB

- El conector de MongoDB actual es compatible con la versión 3.1 o posteriores de Spark y con la versión 4.0 o posteriores de MongoDB.
- Debido a la actualización del conector, se modificaron los nombres de algunas propiedades. Por ejemplo, el nombre de la propiedad de URI se cambió a `connection.uri`. Para obtener más información sobre las opciones actuales, consulte el blog sobre el conector de Spark de MongoDB.
- El uso de la versión 4.0 de MongoDB alojada por Amazon DocumentDB presenta algunas diferencias funcionales. Para obtener más información, consulte estos temas:
  - Functional Differences: Amazon DocumentDB and MongoDB (Diferencias funcionales: Amazon DocumentDB y MongoDB)
  - API, operaciones y tipos de datos de MongoDB compatibles.
- La opción “partitioner” se restringe a `ShardedPartitioner`, `PaginateIntoPartitionsPartitioner` y `SinglePartitionPartitioner`. No puede

usar los valores predeterminados `SamplePartitioner` y `PaginateBySizePartitioner` para Amazon DocumentDB porque el operador de etapas no admite la API de MongoDB. Para obtener más información, consulte Supported MongoDB APIs, Operations, and Data Types (API, operaciones y tipos de datos de MongoDB admitidos).

Delta Lake

- Delta Lake ahora admite viajes en el tiempo en SQL para consultar datos antiguos con facilidad. Con esta actualización, los viajes en el tiempo ahora están disponibles tanto en Spark SQL como a través de la API de DataFrame. Se agregó compatibilidad con la versión actual de TIMESTAMP en SQL.
- Spark 3.3 presenta Trigger.AvailableNow para ejecutar consultas de transmisión como equivalente a `Trigger.Once` para las consultas por lotes. Esta compatibilidad también está disponible cuando se utilizan tablas Delta como origen de transmisión.
- Compatibilidad con SHOW COLUMNS para devolver la lista de columnas de una tabla.
- Compatibilidad con DESCRIBE DETAIL en la API de DeltaTable de Python y Scala. Recupera información detallada sobre una tabla Delta mediante la API de DeltaTable o Spark SQL.
- Compatibilidad para devolver métricas de operaciones de los comandos Delete, Merge y Update de SQL. Anteriormente, estos comandos de SQL devolvían un DataFrame vacío y ahora devuelven un DataFrame con métricas útiles sobre la operación hecha.
- Optimice las mejoras de rendimiento:
  - Defina la opción de configuración `spark.databricks.delta.optimize.repartition.enabled=true` para usar `repartition(1)` en lugar de `coalesce(1)` en el comando Optimize para obtener un mejor rendimiento al compactar muchos archivos pequeños.
  - Rendimiento mejorado mediante el uso de un enfoque basado en colas para paralelizar los trabajos de compactación.
- Otros cambios notables:
  - Compatibilidad con el uso de variables en los comandos de SQL VACUUM y OPTIMIZE.
  - Mejoras para CONVERT TO DELTA con tablas de catálogo que incluyen lo siguiente:
    - Rellenado automático del esquema de particiones a partir del catálogo cuando no se proporciona.
    - Uso de la información de particiones del catálogo para buscar los archivos de datos que se van a confirmar en lugar de analizar el directorio por completo. En lugar de enviar todos los

archivos de datos del directorio de tablas, solo se confirmarán los archivos de datos de los directorios de las particiones activas.

- Compatibilidad con las lecturas por lotes de cambio de fuente de datos (CDF) en las tablas habilitadas para la asignación de columnas cuando no se utilizaron DROP COLUMN y RENAME COLUMN. Para obtener más información, consulte la documentación de Delta Lake.
- Mejora del rendimiento del comando Update al habilitar la eliminación de esquemas en la primera aprobación.

Apache Iceberg

- Se agregaron varias mejoras de rendimiento para la planificación de análisis y las consultas de Spark.
- Se agregó un cliente de catálogo REST común que usa confirmaciones basadas en cambios para resolver conflictos de confirmación en el lado del servicio.
- Se admite la sintaxis `AS OF` para las consultas de viajes en el tiempo de SQL.
- Se agregó compatibilidad de combinación en lectura para las consultas MERGE y UPDATE.
- Se agregó compatibilidad para reescribir particiones con el orden Z.
- Se agregaron una especificación y una implementación para Puffin, un formato para estadísticas y blobs de índice grandes, como bocetos de Theta o filtros de Bloom.
- Se agregaron nuevas interfaces para consumir datos de forma incremental (análisis de registros de cambios y anexos).
- Se agregó compatibilidad con operaciones masivas y lecturas por intervalos en las interfaces de FileIO.
- Se agregaron más tablas de metadatos para mostrar los archivos eliminados en el árbol de metadatos.
- El comportamiento de eliminación de tabla cambió. En Iceberg 0.13.1, la ejecución de `DROP TABLE` elimina la tabla del catálogo, así como su contenido. En Iceberg 1.0.0, `DROP TABLE` solo elimina la tabla del catálogo. Para eliminar el contenido de la tabla, utilice `DROP TABLE PURGE`.
- Las lecturas vectorizadas de Parquet están habilitadas de forma predeterminada en Iceberg 1.0.0. Si desea deshabilitar las lecturas vectorizadas, establezca el valor de `read.parquet.vectorization.enabled` en `false`.

Oracle

Se hicieron pequeños cambios.

MySQL

Se hicieron pequeños cambios.

Amazon Redshift

AWS Glue 4.0 incluye un nuevo conector de Amazon Redshift con un nuevo controlador JDBC. Para obtener información sobre las mejoras y sobre cómo llevar a cabo la migración desde versiones anteriores de AWS Glue, consulte the section called “Conexiones Redshift”.

Apéndice A: actualizaciones de dependencias importantes

Las siguientes son las actualizaciones de dependencias:

Dependencia	Versión en AWS Glue 4.0	Versión en AWS Glue 3.0	Versión en AWS Glue 2.0	Versión en AWS Glue 1.0
Spark	3.3.0-amzn-1	3.1.1-amzn-0	2.4.3	2.4.3
Hadoop	3.3.3-amzn-0	3.2.1-amzn-3	2.8.5-amzn-5	2.8.5-amzn-1
Scala	2.12	2.12	2.11	2.11
Jackson	2.13.3	2.10.x	2.7.x	2.7.x
Hive	2.3.9-amzn-2	2.3.7-amzn-4	1.2	1.2
EMRFS	2.54.0	2.46.0	2.38.0	2.30.0
Json4s	3.7.0-M11	3.6.6	3.5.x	3.5.x
Arrow	7.0.0	2.0.0	0.10.0	0.10.0
Cliente del Catálogo de datos de AWS Glue	3.7.0	3.0.0	1.10.0	N/A

Dependencia	Versión en AWS Glue 4.0	Versión en AWS Glue 3.0	Versión en AWS Glue 2.0	Versión en AWS Glue 1.0
Python	3.10	3.7	2.7 y 3.6	2.7 y 3.6
Boto	1.26	1.18	1.12	N/A

Apéndice B: actualizaciones de controladores JDBC

Las siguientes son las actualizaciones de controladores JDBC:

Controlador	Versión del controlador JDBC en las versiones de AWS Glue anteriores	Versión del controlador JDBC en AWS Glue 3.0	Versión del controlador JDBC en AWS Glue 4.0
MySQL	5.1	8.0.23	8.0.23
Microsoft SQL Server	6.1.0	7.0.0	9.4.0
Oracle Database	11.2	21.1	21.7
PostgreSQL	42.1.0	42.2.18	42.3.6
MongoDB	2.0.0	4.0.0	4.7.2
Amazon Redshift	redshift-jdbc41-1.2.12.1017	redshift-jdbc41-1.2.12.1017	redshift-jdbc42-2.1.0.16

Apéndice C: actualizaciones del conector

A continuación se indican las actualizaciones del conector:

Controlador	Versión del conector en AWS Glue 3.0	Versión del conector en AWS Glue 4.0
MongoDB	3.0.0	10.0.4

Controlador	Versión del conector en AWS Glue 3.0	Versión del conector en AWS Glue 4.0
Hudi	0.10.1	0.12.1
Delta Lake	1.0.0	2.1.0
Iceberg	0.13.1	1.0.0
DynamoDB	1.11	1.12

Migración de AWS Glue para Ray de la vista previa al entorno de tiempo de ejecución de Ray2.4

> **Warning**
>
> Cuando guarde un trabajo de AWS Glue para Ray (vista previa) en AWS Glue Studio, se actualizará automáticamente al entorno de tiempo de ejecución de `Ray2.4`. Si tiene problemas de compatibilidad con el script, póngase en contacto con el servicio de asistencia.

Debería migrar los trabajos de AWS Glue que se crearon durante AWS Glue para Ray (vista previa) a AWS Glue para Ray. Esto implicará algunos cambios simultáneos en la configuración del trabajo.

- En el campo `Runtime`, indique el valor de tiempo de ejecución de `Ray2.4`. Esto actualizará la versión subyacente de Ray de la 2.0.0 a la 2.4.0.
- Algunas bibliotecas de Python incluidas de forma predeterminada en la vista previa ya no se proporcionan. Si el trabajo utiliza el AWS SDK para pandas (awswrangler), dask, modin o pymars, tendrá que incluirlos como bibliotecas adicionales. Para obtener más información sobre la inclusión de bibliotecas de Python adicionales, consulte the section called “Módulos de Python adicionales para trabajos de Ray”.
- Si utiliza el parámetro `--additional-python-modules`, los parámetros utilizados para admitir este flujo de trabajo se han dividido en `--pip-install` y `--s3-py-modules`. Para obtener más información sobre estos parámetros, consulte the section called “Módulos de Python adicionales para trabajos de Ray”.

- Si utiliza el parámetro `--auto-scaling-ray-min-workers`, se ha cambiado el nombre a `--min-workers`.

Política de compatibilidad de versiones de AWS Glue

AWS Glue es un servicio de integración de datos sin servidor que facilita la detección, preparación y combinación de datos para el análisis, machine learning y desarrollo de aplicaciones. Un trabajo de AWS Glue contiene la lógica empresarial que realiza el trabajo de integración de datos en AWS Glue. Existen dos tipos de trabajos en AWS Glue: Spark (por lotes y streaming), Ray y intérprete de comandos de Python. Al definir un trabajo, debe especificar la versión de AWS Glue que configura las versiones en el entorno en tiempo de ejecución de Spark, Ray o Python subyacente. Por ejemplo: un trabajo de AWS Glue de la versión 2.0 de Spark admite Spark 2.4.3 y Python 3.7.

Política de soporte

Ocasionalmente, AWS Glue suspende la compatibilidad con versiones anteriores de AWS Glue. Sin embargo, los trabajos que se ejecutan en versiones obsoletas ya no cuentan con el respaldo del soporte técnico. AWS Glue ya no hará revisiones de seguridad ni otras actualizaciones de las versiones obsoletas. AWS Glue tampoco estará sujeto a los SLA cuando los trabajos se ejecuten en versiones obsoletas.

Cuando ya no se ofrezca compatibilidad con la versión 2.0 o una versión posterior de AWS Glue, no podrá crear trabajos, solo tendrá permitido editarlos o ejecutarlos.

Las siguientes versiones de AWS Glue han alcanzado o están programadas para el fin del soporte: El fin del soporte comienza a medianoche (huso horario del Pacífico) en la fecha especificada.

Tipo	Versión de Glue	Fin del soporte
Spark	Spark 2.2, Scala 2 (Glue versión 0.9)	1/6/2022
Spark	Spark 2.2, Python 2 (Glue versión 0.9)	1/6/2022
Spark	Spark 2.4, Python 2 (Glue versión 1.0)	1/6/2022

Tipo	Versión de Glue	Fin del soporte
Spark	Spark 2.4, Python 3 (Glue versión 1.0)	30/9/2022
Spark	Spark 2.4, Scala 2 (Glue versión 1.0)	30/9/2022
Spark	Glue versión 2.0:	1/31/2024
Tipo	Versión de Python	Fin del soporte
Shell de Python	Python 2 (Glue versión 1.0)	1/6/2022
Tipo	Versión del bloc de notas	Fin del soporte
Punto de enlace de desarrollo	Bloc de notas de Zeppelin	30/9/2022

AWS recomienda encarecidamente migrar los trabajos a versiones compatibles.

Para obtener información sobre la migración de los trabajos de Spark a la versión más reciente de AWS Glue, consulte Migración de trabajos de AWS Glue a la versión 4.0 de AWS Glue.

Para migrar los trabajos de shell de Python a la versión más reciente de AWS Glue:

- En la consola, elija `Python 3 (Glue Version 4.0)`.
- En la API CreateJob o UpdateJob, establezca el parámetro `GlueVersion` en `2.0` y `PythonVersion` en `3`, en el parámetro `Command`. La `GlueVersion` configuración no afecta al comportamiento de los trabajos del intérprete de comandos de Python, por lo que el incremento de `GlueVersion` no supone ninguna ventaja.
- Necesita que el script de su trabajo sea compatible con Python 3.

> **Note**
>
> Todas las regiones de AWS que se lanzaron antes del lanzamiento de la región de Yakarta (Indonesia) (ap-southeast-3) en agosto de 2022 tienen una lista de clientes permitidos a los que se permite ejecutar trabajos de la versión 0.9/1.0 de AWS Glue. En estas regiones antiguas, puede crear un trabajo con un valor nulo y, de forma predeterminada, será

la versión 0.9/1.0, según la región. En el caso de las regiones de AWS que se lancen posteriormente, debe configurar la versión AWS Glue de forma explícita en la API. AWS Glue ya no acepta un parámetro nulo. Si pasa 0.9 o 1.0 al parámetro, aparece el error “La versión 0.9 (o) 1.0 de Glue no es compatible”.

Trabajar con Spark jobs en AWS Glue

Proporciona información sobre los trabajos AWS Glue de ETL de Spark.

Temas

- Parámetros de los trabajos de AWS Glue
- AWS Glue Spark y PySpark empleos
- Trabajos ETL de streaming en AWS Glue
- Coincidencia de registros con FindMatches de AWS Lake Formation
- Migrar programas de Apache Spark a AWS Glue

Parámetros de los trabajos de AWS Glue

Al crear un trabajo de AWS Glue, estableces algunos campos estándar, como `Role` y`WorkerType`. Puede proporcionar información de configuración adicional a través de los campos `Argument` (Parámetros del trabajo en la consola). En estos campos, puede proporcionar a los trabajos de AWS Glue los argumentos (parámetros) enumerados en este tema. Para obtener más información sobre la API AWS Glue Job, consultethe section called “Trabajos”.

Configuración de los parámetros del trabajo

Puede configurar un trabajo a través de la consola, en la pestaña Job details (Detalles del trabajo), en el encabezado Job Parameters (Parámetro del trabajo). También puede configurar un trabajo AWS CLI mediante la configuración `DefaultArguments` o `NonOverridableArguments` en un trabajo, o configurando `Arguments` la ejecución de un trabajo. Los argumentos establecidos en el trabajo se pasarán cada vez que se ejecute el trabajo, mientras que los argumentos establecidos en la ejecución del trabajo solo se pasarán para esa ejecución individual.

Por ejemplo, a continuación se muestra la sintaxis para ejecutar un trabajo con `--arguments` para configurar un parámetro de trabajo.

```
$ aws glue start-job-run --job-name "CSV to CSV" --arguments='--scriptLocation="s3://
my_glue/libraries/test_lib.py"'
```

Acceder a los parámetros del trabajo

Al escribir scripts de AWS Glue, es posible que desee acceder a los valores de los parámetros del trabajo para modificar el comportamiento de su propio código. Proporcionamos métodos auxiliares para hacerlo en nuestras bibliotecas. Estos métodos resuelven los valores de los parámetros de ejecución del trabajo que anulan los valores de los parámetros del trabajo. Al resolver los parámetros establecidos en varios lugares, el trabajo `NonOverridableArguments` anulará el trabajo ejecutado `Arguments`, que anulará el trabajo `DefaultArguments`.

En Python:

En los trabajos de Python, proporcionamos una función llamada `getResolvedParameters`. Para obtener más información, consulte the section called “getResolvedOptions”. Los parámetros del trabajo están disponibles en la variable `sys.argv`.

En Scala:

En los trabajos de Scala, proporcionamos un objeto llamado `GlueArgParser`. Para obtener más información, consulte the section called “GlueArgParser”. Los parámetros del trabajo están disponibles en la variable `sysArgs`.

Referencia de parámetros de trabajo

AWS Glue reconoce los siguientes nombres de argumentos que se pueden utilizar para configurar el entorno de scripts para los trabajos y las ejecuciones de trabajos:

`--additional-python-modules`

Lista delimitada por comas que representa un conjunto de paquetes de Python que se deben instalar. Puede instalar los paquetes desde PyPI o proporcionar una distribución personalizada. Una entrada de paquete de PyPI tendrá el formato *`package`*==*`version`*, con el nombre y la versión del paquete de destino de PyPI. Una entrada de distribución personalizada es la ruta de S3 a la distribución.

Las entradas utilizan la coincidencia de versiones de Python para hacer coincidir el paquete y la versión. Eso significa que tendrá que usar dos signos igual: ==. Existen otros operadores de coincidencia de versiones; para obtener más información, consulte PEP 440.

Para pasar las opciones de instalación del módulo a `pip3`, utilice el parámetro --python-modules-installer-option.

`--auto-scale-within-microbatch`

El valor predeterminado es false. Este parámetro solo se puede usar para los trabajos de transmisión de AWS Glue, que procesan los datos de transmisión en una serie de microlotes, y el escalado automático debe estar habilitado. Al establecer este valor en falso, calcula la media móvil exponencial de la duración del lote de los microlotes completados y lo compara con el tamaño de la ventana para determinar si se debe escalar verticalmente o reducir verticalmente el número de ejecutores. El escalado solo se produce cuando se completa un microlote. Al establecer este valor en verdadero, durante un microlote, escala verticalmente cuando el número de tareas de Spark permanece igual durante 30 segundos o cuando el procesamiento por lotes actual es superior al tamaño de la ventana. El número de ejecutores disminuirá si un ejecutor ha estado inactivo durante más de 60 segundos o si la media móvil exponencial de la duración del lote es baja.

`--class`

La clase de Scala que sirve de punto de entrada del script de Scala. Esto solo se aplica si `--job-language` está establecido en `scala`.

`--continuous-log-conversionPattern`

Especifica un patrón de registro de conversión personalizado para un trabajo habilitado para el registro continuo. El patrón de conversión solo se aplica a los registros de controlador y de ejecutor. No afecta a la barra de progreso de AWS Glue.

`--continuous-log-logGroup`

Especifica un nombre de grupo de CloudWatch registros de Amazon personalizado para un trabajo habilitado para el registro continuo.

`--continuous-log-logStreamPrefix`

Especifica un prefijo de flujo de CloudWatch registro personalizado para un trabajo habilitado para el registro continuo.

`--customer-driver-env-vars` y **`--customer-executor-env-vars`**

Estos parámetros establecen variables de entorno en sistemas operativos de manera respectiva para cada trabajador (controlador o ejecutor). Puedes usar estos parámetros al crear plataformas y marcos personalizados sobre AWS Glue, para que tus usuarios puedan escribir trabajos sobre él. Al activar estos dos marcadores, podrá establecer distintas variables de entorno para el

controlador y para el ejecutor respectivamente sin tener que inyectar la misma lógica en el propio script del trabajo.

Ejemplo de uso

A continuación se muestra un ejemplo de la utilización de estos parámetros:

```
"–customer-driver-env-vars", "CUSTOMER_KEY1=VAL1,CUSTOMER_KEY2=\"val2,val2 val2\"",
"–customer-executor-env-vars", "CUSTOMER_KEY3=VAL3,KEY4=VAL4"
```

El establecimiento de estos en el argumento del trabajo equivale a la ejecución de los siguientes comandos:

En el controlador:

- exportar CUSTOMER_KEY1=VAL1
- exportar CUSTOMER_KEY2="val2,val2 val2"

En el ejecutor:

- exportar CUSTOMER_KEY3=VAL3

Luego, en el propio script de trabajo, se pueden recuperar las variables de entorno con `os.environ.get("CUSTOMER_KEY1")` o `System.getenv("CUSTOMER_KEY1")`.

Sintaxis de uso obligatorio

Controle los siguientes estándares al definir las variables del entorno:

- Cada clave debe contener el `CUSTOMER_ prefix`.

  Por ejemplo: para `"CUSTOMER_KEY3=VAL3,KEY4=VAL4"`, se ignorará `KEY4=VAL4` y no se establecerá.
- Cada par de clave y valor debe estar delimitado por una coma simple.

  Por ejemplo: `"CUSTOMER_KEY3=VAL3,CUSTOMER_KEY4=VAL4"`
- Si el valor contiene espacios o comas, debe encerrarse entre comillas.

  Por ejemplo: `CUSTOMER_KEY2=\"val2,val2 val2\"`

Esta sintaxis modela con precisión los estándares de configuración de las variables de entorno de bash.

`--datalake-formats`

Compatible con AWS Glue 3.0 y versiones posteriores.

Especifica el marco del lago de datos que se va a utilizar. AWS Glue añade los archivos JAR necesarios para los marcos que especifique en el`classpath`. Para obtener más información, consulte Uso de marcos de lagos de datos con trabajos de ETL de AWS Glue.

Puede especificar uno o varios de los siguientes valores, separados por una coma:

- `hudi`
- `delta`
- `iceberg`

Por ejemplo, pase el siguiente argumento para especificar los tres marcos.

```
'--datalake-formats': 'hudi,delta,iceberg'
```

`--disable-proxy-v2`

Inhabilite el proxy del servicio para permitir las llamadas de AWS servicio a Amazon S3 y que se AWS Glue originen desde su script a través de su VPC. CloudWatch Para obtener más información, consulte Configuración de las llamadas de AWS para que pasen a través de su VPC. Para deshabilitar el proxy del servicio, defina el valor de este parámetro en `true`.

`--enable-auto-scaling`

Cuando este valor se configura en `true`, se activa el escalado automático y la facturación por proceso de trabajo.

`--enable-continuous-cloudwatch-log`

Permite un registro continuo en tiempo real de los trabajos de AWS Glue. Puede ver los registros de trabajo de Apache Spark en CloudWatch tiempo real.

`--enable-continuous-log-filter`

Especifica un filtro estándar (`true`) o ningún filtro (`false`) al crear o editar un trabajo habilitado para el registro continuo. Si se elige Standard filter (Filtro estándar), se filtran los mensajes de registro de estado de Apache Hadoop YARN y de controlador/ejecutor de Apache Hadoop YARN no útiles. Si se elige sin filtro, se verán todos los mensajes de registro.

`--enable-glue-datacatalog`

Le permite utilizar el catálogo de datos de AWS Glue como un metaalmacén de Apache Spark Hive. Para habilitar esta característica, establezca el valor como `true`.

`--enable-job-insights`

Permite una supervisión adicional del análisis de errores con información sobre la ejecución de los trabajos de AWS Glue. Para obtener más detalles, consulte the section called “Supervisión con Información de ejecuciones de trabajos de AWS Glue”. De forma predeterminada, el valor se establece en `true` y la información de ejecución de trabajos está habilitada.

Esta opción está disponible para AWS Glue versión 2.0 y 3.0.

`--enable-metrics`

Permite recopilar métricas para generar perfiles de trabajo en la ejecución de este trabajo. Estas métricas están disponibles en la AWS Glue consola y en la CloudWatch consola de Amazon. El valor de este parámetro no es relevante. Para activar esta característica, puede proporcionar a este parámetro cualquier valor, pero se recomienda `true` para mayor claridad. Para deshabilitar esta característica, elimine este parámetro de la configuración de su trabajo.

`--enable-observability-metrics`

Permite un conjunto de métricas de observabilidad para generar información sobre lo que sucede dentro de cada trabajo que se ejecuta en la página Job Runs Monitoring en la AWS Glue consola y la Amazon CloudWatch consola. Para habilitar esta característica, establezca el valor como true (verdadero). Para deshabilitar esta característica, defínala en `false` o elimine este parámetro de la configuración de su trabajo.

`--enable-rename-algorithm-v2`

Establece la versión del algoritmo de cambio de nombre de EMRFS a la versión 2. Cuando un trabajo de Spark utiliza el modo de sobrescritura de partición dinámica, existe la posibilidad de que se cree una partición duplicada. Por ejemplo, puede tener como resultado una partición duplicada como `s3://bucket/table/location/p1=1/p1=1`. Aquí, P1 es la partición que se está sobrescribiendo. El algoritmo de cambio de nombre versión 2 corrige este problema.

Esta opción solo se encuentra disponible en la versión 1.0 de AWS Glue.

`--enable-s3-parquet-optimized-committer`

Habilita el confirmador optimizado para S3 de EMRFS de forma que puedan escribirse datos de Parquet en Amazon S3. Puede suministrar el par parámetro/valor a través de la consola de AWS

Glue cuando cree o actualice un trabajo de AWS Glue. Al establecer el valor en **`true`**, se habilita el confirmador. De forma predeterminada, la marca está activada en AWS Glue 3.0 y desactivada en AWS Glue 2.0.

Para obtener más información, consulte este artículo sobre el uso del confirmador optimizado para S3 de EMRFS.

`--enable-spark-ui`

Cuando se establece en `true`, se activa la característica que permite utilizar la interfaz de usuario de Spark para supervisar y depurar trabajos de ETL de AWS Glue.

`--executor-cores`

Número de tareas de Spark que se pueden ejecutar en paralelo. Esta opción es compatible con AWS Glue 3.0+. El valor no debe superar el doble de la cantidad de vCPU en el tipo de trabajador, es decir, 8 en `G.1X`, 16 en `G.2X`, 32 en `G.4X` y 64 en `G.8X`. Debe tener cuidado al actualizar esta configuración, ya que podría afectar al rendimiento laboral porque el aumento del paralelismo entre las tareas provoca presión en la memoria y el disco y podría limitar los sistemas de origen y destino (por ejemplo, provocaría más conexiones simultáneas en Amazon RDS).

`--extra-files`

Las rutas de Amazon S3 a archivos adicionales, por ejemplo, archivos de configuración que AWS Glue copia en el directorio de trabajo de su script antes de ejecutarlo. Varios valores deben ser rutas completas separadas por una coma (`,`). Solo se admiten archivos individuales, no una ruta de directorio. Esta opción no es compatible con los tipos de trabajo de intérprete de comandos de Python.

`--extra-jars`

Las rutas de Amazon S3 a archivos `.jar` de Java adicionales que AWS Glue agrega al classpath Java antes de ejecutar el script. Varios valores deben ser rutas completas separadas por una coma (`,`).

`--extra-py-files`

Las rutas de Amazon S3 a módulos de Python adicionales que AWS Glue agrega a la ruta de Python antes de ejecutar el script. Varios valores deben ser rutas completas separadas por una coma (`,`). Solo se admiten archivos individuales, no una ruta de directorio.

`--job-bookmark-option`

Controla el comportamiento de un marcador de trabajo. Se pueden establecer los siguientes valores de opciones.

--job-bookmark-option Valor	Descripción
`job-bookmark-enable`	Realizar un seguimiento de los datos procesados anteriormente. Cuando se ejecuta un flujo de trabajo, procesar los datos nuevos desde el último punto de comprobación.
`job-bookmark-disable`	Procesar siempre todo el conjunto de datos. Usted es responsable de administrar el resultado de las ejecuciones de trabajos anteriores.
`job-bookmark-pause`	Procesar los datos incrementales desde la última ejecución correcta o los datos en el rango identificado por las siguientes subopciones, sin actualizar el estado del último marcador. Usted es responsable de administr ar el resultado de las ejecuciones de trabajos anteriore s. Las dos subopciones son las siguientes: • **`job-bookmark-from`** `<from-value>` es el ID de ejecución que representa todas las entradas que se han procesado hasta la última ejecución correcta antes de la misma e incluye el ID de ejecución especificado. La entrada correspondiente se ignora. • **`job-bookmark-to`** `<to-value>` es el ID de ejecución que representa todas las entradas que se han procesado hasta la última ejecución correcta antes de la misma e incluye el ID de ejecución especificado. El trabajo procesa la entrada correspon diente, excluida la entrada identificada por `<from-value>` . Cualquier entrada posterior a esta entrada también se excluye para el procesamiento. El estado del marcador de trabajo no se actualiza cuando se especifica este conjunto de opciones.

--job-bookmark-option Valor	Descripción
	Las subopciones son opcionales. Sin embargo, cuando se utilizan, deben proporcionarse ambas subopciones.

Por ejemplo, para habilitar un marcador de trabajo, transfiera el argumento siguiente.

```
'--job-bookmark-option': 'job-bookmark-enable'
```

`--job-language`

El lenguaje de programación del script. Este valor debe ser `scala` o `python`. Si no está presente este parámetro, el valor predeterminado es `python`.

`--python-modules-installer-option`

Una cadena de texto simple que define las opciones que se van a pasar a `pip3` al instalar módulos con `--additional-python-modules`. Proporcione opciones como lo haría en la línea de comandos, separadas por espacios y prefijadas por guiones. Para obtener más información acerca del uso de una VPC, consulte the section called “Instalación de módulos de Python adicionales con pip en AWS Glue 2.0+”.

> **Note**
>
> Esta opción no es compatible con los trabajos de AWS Glue cuando se utiliza Python 3.9.

`--scriptLocation`

La ubicación de Amazon Simple Storage Service (Amazon S3) donde se encuentra su script de ETL (en formato `s3://path/to/my/script.py`). Este parámetro invalida una ubicación del script establecida en el objeto `JobCommand`.

`--spark-event-logs-path`

Especifica una ruta de Amazon S3. Al utilizar la característica de monitoreo de la interfaz de usuario de Spark, AWS Glue vacía los registros de eventos de Spark a esta ruta de Amazon S3 cada 30 segundos, en un bucket que se puede utilizar como directorio temporal para almacenar eventos de la interfaz de usuario de Spark.

`--TempDir`

Especifica una ruta de Amazon S3 a un bucket que se puede utilizar como directorio temporal para el trabajo.

Por ejemplo, para establecer un directorio temporal, transfiera el argumento siguiente.

```
'--TempDir': 's3-path-to-directory'
```

> **Note**
>
> AWS Glue crea un bucket temporal para los trabajos si aún no existe ningún bucket en una región. Este bucket podría permitir el acceso público. Puede modificar el bucket en Amazon S3 para establecer el bloqueo del acceso público o eliminar el bucket más tarde, después de que se hayan completado todos los trabajos de esa región.

`--use-postgres-driver`

Al establecer este valor en `true`, se prioriza el controlador JDBC de Postgre en la ruta de clases para evitar un conflicto con el controlador JDBC de Amazon Redshift. La opción solo se encuentra disponible en la versión 2.0 de AWS Glue.

`--user-jars-first`

Al establecer este valor en `true`, se priorizan los archivos JAR adicionales del cliente en el classpath. Esta opción solo se encuentra disponible en AWS Glue versión 2.0 o posterior.

`--conf`

Controla los parámetros de configuración de Spark. Es para casos de uso avanzados.

`--encryption-type`

Parámetro heredado. El comportamiento correspondiente debe configurarse mediante configuraciones de seguridad. Para obtener más información sobre las configuraciones de seguridad, consulte the section called “Cifrado de datos escritos por AWS Glue”.

AWS Glue utiliza los siguientes argumentos a nivel interno y el usuario nunca debe utilizarlos:

- `--debug`: interno de AWS Glue. No lo establezca.

- `--mode`: interno de AWS Glue. No lo establezca.
- `--JOB_NAME`: interno de AWS Glue. No lo establezca.
- `--endpoint`: interno de AWS Glue. No lo establezca.

AWS Glue admite el inicio de un entorno con el módulo `site` de Python que utiliza `sitecustomize` para realizar personalizaciones específicas del sitio. Se recomienda iniciar sus propias funciones de inicialización únicamente para casos de uso avanzados y, si se hace todo lo posible, se admite en la versión 4.0 AWS Glue.

El prefijo de la variable de entorno, `GLUE_CUSTOMER`, está reservado para el uso de los clientes.

AWS Glue Spark y PySpark empleos

Las siguientes secciones proporcionan información sobre AWS Glue Spark y los PySpark trabajos.

Temas

- Agregar trabajos de Spark y PySpark en AWS Glue
- Seguimiento de los datos procesados mediante marcadores de trabajo
- Complemento de mezclas aleatorias de AWS Glue Spark con Amazon S3
- Supervisión de trabajos de Spark de AWS Glue

Agregar trabajos de Spark y PySpark en AWS Glue

En las siguientes secciones, se ofrece información acerca de cómo agregar trabajos Spark y PySpark en AWS Glue.

Temas

- Configurar las propiedades de los trabajos de Spark en AWS Glue
- Edición de scripts de Spark en la consola de AWS Glue
- Trabajos (heredado)

Configurar las propiedades de los trabajos de Spark en AWS Glue

Un trabajo de AWS Glue encapsula un script que se conecta a los datos de origen, los procesa y, a continuación, los escribe en el destino de datos. Normalmente, un trabajo ejecuta scripts de

extracción, transformación y carga (ETL). Los trabajos también pueden ejecutar scripts de Python de uso general (trabajos de shell de Python). Los desencadenadores de AWS Glue pueden iniciar trabajos en función de un programa o evento, o bajo demanda. Puede monitorear las ejecuciones de trabajos para comprender las métricas de tiempo de ejecución como el estado de realización, la duración y la hora de inicio.

Puede utilizar los scripts que genera AWS Glue o puede utilizar sus propios scripts. Con un esquema de origen y una ubicación o esquema de destino, el generador de AWS Glue código puede crear automáticamente un script de la API (PySpark) de Apache Spark. Puede utilizar este script como base y editarlo para satisfacer sus objetivos.

AWS Glue puede escribir archivos de salida en varios formatos de datos, como JSON, CSV, ORC (Almacenamiento de filas en columnas optimizado), Apache Parquet y Apache Avro. En algunos formatos de datos, se pueden escribir formatos de compresión comunes.

Hay tres tipos de trabajos en AWS Glue: Spark, ETL de streaming y shell de Python.

- Un trabajo de Spark se ejecuta en un entorno de Apache Spark gestionado por AWS Glue. Procesa los datos en lotes.
- Un trabajo ETL de streaming es similar a un trabajo de Spark, excepto que realiza ETL en las transmisiones de datos. Utiliza el marco Apache Spark Structured Streaming. Algunas características de trabajo de Spark no están disponibles para los trabajos ETL de streaming.
- Un trabajo de shell de Python ejecuta scripts de Python como un shell y soporta una versión de Python según la versión de AWS Glue que esté utilizando. Estos trabajos pueden utilizarse para programar y ejecutar tareas que no requieren un entorno de Apache Spark.

Definición de propiedades de trabajo para trabajos de Spark

Cuando define su trabajo en la consola de AWS Glue, proporciona valores de propiedades para controlar el entorno en tiempo de ejecución de AWS Glue.

En la siguiente lista se describen las propiedades de un trabajo de Spark. Para ver las propiedades de un trabajo de shell de Python, consulte Definición de las propiedades de trabajos de shell de Python. Para obtener información sobre las propiedades de un trabajo ETL de streaming, consulte the section called “Definición de propiedades de trabajo para un trabajo ETL de streaming”.

Las propiedades se muestran en el orden en que aparecen en el asistente Add job (Agregar trabajo) en la consola de AWS Glue.

Nombre

Proporciona una cadena UTF-8 con una longitud máxima de 255 caracteres

Descripción

Brinde una descripción adicional de hasta 2048 caracteres.

Rol de IAM

Especifique el rol de IAM que se utiliza para dar una autorización sobre los recursos que se utilizan para ejecutar el trabajo y obtener acceso a los almacenes de datos. Para obtener más información acerca de los permisos para ejecutar trabajos en AWS Glue, consulte Gestión de identidad y acceso para AWS Glue.

Tipo

El tipo de trabajo de ETL. Este se establece de manera automática según el tipo de origen de datos que se seleccione.

- Spark ejecuta un script de ETL de Apache Spark con el comando de trabajo `glueetl`.
- Spark Streaming ejecuta un script de ETL de streaming de Apache Spark con el comando de trabajo `gluestreaming`. Para obtener más información, consulte the section called “Trabajos ETL de streaming”.
- El intérprete de comandos de Python ejecuta un script de Python con el comando de trabajo `pythonshell`. Para obtener más información, consulte Configuración de las propiedades de trabajos del intérprete de comandos de Python en AWS Glue.

versión de AWS Glue

La versión de AWS Glue determina las versiones de Apache Spark y Python que están disponibles para el trabajo, como se especifica en la tabla a continuación.

Versión de AWS Glue	Versiones de Spark y Python admitidas
4.0	• Spark 3.3.0 • Python 3.10
3.0	• Spark 3.1.1 • Python 3.7

Versión de AWS Glue	Versiones de Spark y Python admitidas
2.0	• Spark 2.4.3 • Python 3.7
1.0	• Spark 2.4.3 • Python 2.7 • Python 3.6
0.9	• Spark 2.2.1 • Python 2.7

Tipo de empleado

Están disponibles los siguientes tipos de proceso de trabajo:

Los recursos disponibles para los AWS Glue trabajadores se miden en DPUs. Una DPU es una medida relativa de la potencia de procesamiento que consta de 4 vCPU de capacidad de cómputo y 16 GB de memoria.

- **`G.1X`**: al elegir este tipo, también debe proporcionar un valor para Número de empleados. Cada trabajador se asigna a 1 DPU (4 vCPU, 16 GB de memoria) con un disco de 84 GB (aproximadamente 34 GB libres). Recomendamos este tipo de trabajador para cargas de trabajo como transformaciones de datos, uniones y consultas, ya que ofrece una forma escalable y rentable de ejecutar la mayoría de los trabajos.
- **`G.2X`**: al elegir este tipo, también debe proporcionar un valor para Número de empleados. Cada trabajador se asigna a 2 DPU (8 vCPU, 32 GB de memoria) con un disco de 128 GB (aproximadamente 77 GB libres). Recomendamos este tipo de trabajador para cargas de trabajo como transformaciones de datos, uniones y consultas, ya que ofrece una forma escalable y rentable de ejecutar la mayoría de los trabajos.
- **`G.4X`**: al elegir este tipo, también debe proporcionar un valor para Número de empleados. Cada trabajador se asigna a 4 DPU (16 vCPU, 64 GB de memoria) con un disco de 256 GB (aproximadamente 235 GB libres). Recomendamos este tipo de trabajador para los trabajos cuyas cargas de trabajo contengan las transformaciones, agregaciones, uniones y consultas más exigentes. Este tipo de trabajador solo está disponible para los trabajos de ETL de Spark de la AWS Glue versión 3.0 o posteriores en AWS las siguientes regiones: EE.UU. Este (Ohio), EE.UU. Este (Norte de Virginia), EE.UU. Oeste (Oregón), Asia Pacífico (Singapur), Asia

Pacífico (Sídney), Asia Pacífico (Tokio), Canadá (Centro), Europa (Fráncfort), Europa (Irlanda) y Europa (Estocolmo).

- **`G.8X`**: al elegir este tipo, también debe proporcionar un valor para Número de empleados. Cada trabajador se asigna a 8 DPU (32 vCPU, 128 GB de memoria) con un disco de 512 GB (aproximadamente 487 GB libres). Recomendamos este tipo de trabajador para los trabajos cuyas cargas de trabajo contengan las transformaciones, agregaciones, uniones y consultas más exigentes. Este tipo de trabajador solo está disponible para los trabajos ETL de Spark de la AWS Glue versión 3.0 o posteriores, en las mismas AWS regiones compatibles con el tipo de `G.4X` trabajador.
- **`G.025X`**: al elegir este tipo, también debe proporcionar un valor para Número de empleados. Cada trabajador se asigna a 0,25 DPU (2 vCPU, 4 GB de memoria) con un disco de 84 GB (aproximadamente 34 GB libres). Le recomendamos este tipo de proceso de trabajo para trabajos de streaming de bajo volumen. Este tipo de proceso de trabajo solo está disponible para trabajos de streaming con la versión 3.0 de AWS Glue.

Se le cobra una tarifa por hora en función de la cantidad de DPU que se utilizan para ejecutar sus trabajos ETL. Para obtener más información, consulte la página de precios de AWS Glue.

Para trabajos de AWS Glue versión 1.0 o anterior, cuando configura un trabajo con la consola y especifica un Worker type (Tipo de empleado) de Standard (Estándar), se configura la Maximum capacity (Capacidad máxima) y el Number of workers (Número de empleados) se convierte en el valor de Máximum capacity (Capacidad máxima): 1. Si utilizas el AWS Command Line Interface (AWS CLI) o el AWS SDK, puedes especificar el parámetro de capacidad máxima o puedes especificar tanto el tipo de trabajador como el número de trabajadores.

Para trabajos de versión 2.0 de AWS Glue o posterior, no puede especificar una Capacidad máxima. En su lugar, debe especificar un Worker type (Tipo de empleado) y el Number of workers (Número de empleados).

Idioma

El código del script de ETL define la lógica del trabajo. El script se puede codificar en Python o en Scala. Puede elegir si AWS Glue genera el script que el flujo de trabajo ejecuta o bien si usted mismo lo proporciona. Proporcione el nombre del script y la ubicación en Amazon Simple Storage Service (Amazon S3). Compruebe que no haya un archivo con el mismo nombre que el directorio de script en la ruta. Para obtener más información acerca de cómo usar scripts, consulte Guía de programación de AWS Glue.

Número de trabajadores requerido

Para la mayoría de los tipos de trabajador, debe especificar el número de trabajadores que se asigna cuando se ejecuta el trabajo.

Job bookmark (Marcador de flujo de trabajo)

Especifique cómo AWS Glue procesará la información de estado cuando se ejecute el flujo de trabajo. Puede hacer que recuerde datos procesados previamente, actualice la información de estado o no tenga en cuenta dicha información. Para obtener más información, consulte the section called “Seguimiento de los datos procesados mediante marcadores de trabajo”.

Ejecución flexible

Al configurar un trabajo mediante AWS Studio o la API, puede especificar una clase de ejecución de trabajos estándar o flexible. Sus trabajos pueden tener diferentes grados de prioridad y sensibilidad temporal. La clase de ejecución estándar es ideal para cargas de trabajo urgentes que requieren un inicio rápido de los trabajos y recursos dedicados.

La clase de ejecución flexible es adecuada para trabajos no urgentes, como trabajos de preproducción, pruebas y cargas de datos únicas. Se admiten ejecuciones de trabajos flexibles para trabajos que utilizan AWS Glue versión 3.0 o posterior y tipos de empleados `G.1X` o `G.2X`.

Las ejecuciones de trabajos flexibles se facturan en función del número de empleados que se ejecutan en un momento dado. Se puede agregar o eliminar el número de trabajadores para una ejecución de trabajo flexible en ejecución. En lugar de facturar como un simple cálculo de `Max Capacity * Execution Time`, cada trabajador contribuirá por el tiempo que ejecutó durante la ejecución del trabajo. La factura es la suma de (`Number of DPUs per worker * time each worker ran`).

Para obtener más información, consulta el panel de ayuda de AWS Studio o Jobs yEjecuciones de trabajo.

Número de reintentos

Especifique el número de veces, entre 0 y 10, que AWS Glue debe reiniciar automáticamente el flujo de trabajo si se produce un error. Los trabajos que alcanzan el límite de tiempo de espera no se reinician.

Job timeout (Tiempo de espera de flujo de trabajo)

Establece el tiempo de ejecución máximo en minutos. El valor predeterminado es 2880 minutos (48 horas) para los trabajos por lotes. Cuando el tiempo de ejecución de un trabajo supera este límite, el estado de ejecución de trabajo cambia a `TIMEOUT`.

Los trabajos de streaming deben tener valores de tiempo de espera inferiores a 7 días o 10080 minutos. Si el valor se deja en blanco, el trabajo se reiniciará después de 7 días si no se ha configurado un período de mantenimiento. Si ha configurado un período de mantenimiento, se reiniciará durante el período de mantenimiento transcurridos 7 días.

> **Prácticas recomendadas para los tiempos de espera de trabajo**
>
> Los trabajos se cobran de acuerdo a los tiempos de ejecución. Para evitar costes inesperados, configure los valores de tiempos de espera correctos para el tiempo de ejecución esperado del trabajo.

Propiedades avanzadas

Nombre de archivo del script

Un nombre de script único para el trabajo. El nombre no puede ser Trabajo sin nombre.

Ruta del script

La ubicación del script de Amazon S3. La ruta debe tener el formato `s3://bucket/prefix/path/`. Debe terminar con una barra (`/`) y no debe incluir ningún archivo.

Métricas de trabajo

Activa o desactiva la creación de CloudWatch métricas de Amazon cuando se ejecute este trabajo. Para ver los datos de perfiles, debe habilitar esta opción. Para obtener más información acerca de cómo habilitar y visualizar métricas, consulte Monitorización y depuración de trabajo.

Métricas de observabilidad de trabajos

Activa la creación de CloudWatch métricas de observabilidad adicionales cuando se ejecute este trabajo. Para obtener más información, consulte the section called “Monitorización con métricas de observabilidad de AWS Glue”.

Registro continuo

Activa el registro continuo en Amazon CloudWatch. Si esta opción no está habilitada, los registros solo estarán disponibles después de que se complete el trabajo. Para más información, consulte the section called “Registro continuo para trabajos de AWS Glue”.

Interfaz de usuario de Spark

Habilite el uso de la interfaz de usuario (UI) de Spark para monitorear este trabajo. Para obtener más información, consulte Habilitación de la interfaz de usuario web de Apache Spark para trabajos de AWS Glue.

Ruta de los registros de la interfaz de usuario de Spark

La ruta para escribir registros mientras la interfaz de usuario de Spark está habilitada.

Configuración del registro y el monitoreo de la interfaz de Spark

Seleccione una de las siguientes opciones:

- Estándar: escribe los registros con el ID de ejecución de la AWS Glue tarea como nombre de archivo. Activa la supervisión de la interfaz de usuario de Spark en la AWS Glue consola.
- Heredado: escriba registros con el nombre “spark-application-{marca de tiempo}”. No active el monitoreo de la interfaz de usuario de Spark.
- Estándar y heredado: escriba registros tanto en las ubicaciones estándar como heredadas. Activa la monitorización de la interfaz de usuario de Spark en la AWS Glue consola.

Simultaneidad máxima

Establece el número máximo de ejecuciones simultáneas que están permitidas para este flujo de trabajo. El valor predeterminado de es 1. Se produce un error cuando se llega a este umbral. El valor máximo que puede especificar se controla mediante un límite de servicio. Por ejemplo, si una ejecución anterior de un trabajo se sigue ejecutando cuando una nueva instancia se inicia, es posible que desee devolver un error al evitar dos instancias de la misma ejecución de trabajo de forma simultánea.

Ruta temporaria

Proporcione la ubicación de un directorio de trabajo en Amazon S3 donde los resultados intermedios temporales se escriben cuando AWS Glue ejecuta el script. Compruebe que no haya un archivo con el mismo nombre que el directorio temporal en la ruta. Este directorio se utiliza cuando AWS Glue lee y escribe en Amazon Redshift y mediante determinadas transformaciones de AWS Glue.

> **Note**
>
> AWS Glue crea un bucket temporal para los trabajos si aún no existe un bucket en la región. Este bucket podría permitir el acceso público. Puede modificar el bucket en Amazon S3 para establecer el bloque de acceso público o eliminar el bucket más tarde, después de que se hayan completado todos los trabajos de esa región.

Umbral de notificación de retraso (minutos)

Establece el umbral (en minutos) antes de que se envíe una notificación de retraso. Puede configurar este umbral para enviar notificaciones cuando una ejecución de flujo de trabajo `RUNNING`, `STARTING` o `STOPPING` dure más de un número previsto de minutos.

Configuración de seguridad

Elija una configuración de seguridad de la lista. Una configuración de seguridad especifica cómo se cifran los datos en el destino de Amazon S3: sin cifrado, cifrado del lado del servidor con claves administradas por AWS KMS(SSE-KMS) o claves de cifrado administradas por Amazon S3 (SSE-S3).

Cifrado en el servidor

Si selecciona esta opción, cuando el flujo de trabajo de ETL se escribe en Amazon S3, los datos se cifran en reposo mediante cifrado SSE-S3. Tanto el destino de datos de Amazon S3 como los datos que se escriben en un directorio temporal de Amazon S3 se cifran. Esta opción se pasa como parámetro de trabajo. Para obtener más información, consulte Protecting Data Using Server-Side Encryption with Amazon S3-Managed Encryption Keys (SSE-S3) (Protección de datos mediante el cifrado que se ejecuta en el servidor con las claves de cifrado administradas por Amazon S3 [SSE-S3]) en la Guía del usuario de Amazon Simple Storage Service.

> **Important**
>
> Esta opción se pasa por alto si se especifica una configuración de seguridad.

Use Glue Data Catalog as the Hive metastore (Utilizar el catálogo de datos de Glue como metaalmacén de Hive)

Seleccione esta opción para usar el AWS Glue Data Catalog como metaalmacén de Hive. El rol de IAM que se utiliza para el trabajo debe tener el permiso `glue:CreateDatabase`. Si no existe una base de datos, se crea una denominada “default (predeterminada)” en el Data Catalog.

Conexiones

Seleccione una configuración de VPC para acceder a los orígenes de datos de Amazon S3 ubicados en la nube privada virtual (VPC). En AWS Glue, puede crear y administrar la conexión de red. Para obtener más información, consulte Conexión a datos.

Bibliotecas

La ruta de la biblioteca de Python, la ruta de los archivos JAR dependientes y la ruta de archivos referenciados

Especifique estas opciones si el script las requiere. Puede definir las rutas de Amazon S3 separadas por comas para estas opciones al definir el trabajo. Puede omitir estas rutas al ejecutar el flujo de trabajo. Para obtener más información, consulte Suministro de sus propios scripts personalizados.

Parámetros del flujo de trabajo

Conjunto de pares de clave-valor que se pasan como parámetros de nombres al script. Estos son valores predeterminados que se utilizan cuando se ejecuta el script, pero se pueden invalidar en desencadenadores o cuando se ejecuta el trabajo. Debe agregar el prefijo `--` al nombre de clave; por ejemplo: `--myKey`. Los parámetros del trabajo se transmiten como un mapa cuando se utiliza el AWS Command Line Interface.

Para obtener ejemplos, consulte los parámetros de Python en Suministro y acceso a los parámetros de Python en AWS Glue.

Etiquetas

Etiquete su trabajo con una Clave de etiqueta y un Valor de etiqueta opcional. Una vez que se crean las claves de etiquetas, son de solo lectura. Utilice etiquetas en algunos recursos para que le resulte más fácil organizarlos e identificarlos. Para más información, consulte Etiquetas de AWS en AWS Glue.

Restricciones para trabajos que acceden a tablas administradas por Lake Formation

Tenga en cuenta las siguientes notas y restricciones al crear trabajos que lean o escriban en tablas administradas por AWS Lake Formation:

- Las siguientes funciones no se soportan en trabajos que acceden a tablas con filtros de nivel de celda:
  - Marcadores de trabajo y ejecución delimitada
  - Predicados de inserción
  - Predicados de particiones de catálogo del lado del servidor
  - enableUpdateCatalog

Edición de scripts de Spark en la consola de AWS Glue

Un script contiene el código que extrae los datos de las fuentes, los transforma y los carga en los destinos. AWS Glue ejecuta un script al iniciar un trabajo.

Los scripts de ETL de AWS Glue pueden codificarse en Python o Scala. Los scripts de Python utilizan un lenguaje que es una extensión del dialecto de PySpark Python para los trabajos de extracción, transformación y carga (ETL). El script contiene constructos ampliados para gestionar las transformaciones de ETL. Al generar automáticamente lógica de código fuente para el flujo de trabajo, se crea un script. Puede editar este script o proporcionar su propio script para procesar el flujo de trabajo de ETL.

Para obtener más información acerca de cómo definir y editar scripts en AWS Glue, consulte Guía de programación de AWS Glue.

Bibliotecas o archivos adicionales

Si su script requiere bibliotecas o archivos adicionales, puede especificarlos del modo siguiente:

Ruta de la biblioteca Python

Rutas de Amazon Simple Storage Service (Amazon S3) separadas por comas a las bibliotecas de Python requeridas por el script.

> **Note**
>
> Solo se pueden utilizar bibliotecas Python puras. Todavía no se admiten las bibliotecas que se basan en las extensiones de C, como la biblioteca de análisis de datos Python pandas.

Ruta de archivos JAR dependientes

Rutas de Amazon S3 separadas por comas a archivos JAR que requiere el script.

> **Note**
>
> Actualmente, solo se pueden usar bibliotecas Java o Scala (2.11).

Ruta de archivos a la que se hace referencia

Rutas de Amazon S3 separadas por comas a archivos adicionales (por ejemplo, archivos de configuración) requeridas por el script.

Trabajos (heredado)

Los scripts contienen el código que lleva a cabo el flujo de trabajo de extraer, transformar y cargar (ETL). Puede suministrar su propio script o bien, AWS Glue puede generarlo con su ayuda. Para obtener información sobre la creación de sus propios scripts, consulte Suministro de sus propios scripts personalizados.

Puede editar scripts en la consola de AWS Glue. Cuando edite scripts, podrá añadir orígenes, destinos y transformaciones.

Para editar un script

1. Inicie sesión en la AWS Management Console y abra la consola de AWS Glue en https://console.aws.amazon.com/glue/. Después, seleccione pestaña Jobs (Trabajos).
2. Seleccione un flujo de trabajo de la lista y, a continuación, seleccione Action (Acción), Edit script (Editar script) para abrir el editor de scripts.

   También puede acceder al editor de scripts desde la página de detalles del flujo de trabajo. Elija la pestaña Script, y luego elija Edit script (Editar script).

Editor de scripts

Con el editor de scripts de AWS Glue, puede insertar, modificar y eliminar orígenes, destinos y transformaciones en el script. El editor de scripts muestra el script y un diagrama para ayudarle a visualizar el flujo de datos.

Para crear un diagrama para el script, elija Generate diagram (Genera diagrama). AWS Glue utiliza líneas de anotación en el script que comienzan con # # para representar el diagrama. Para representar correctamente el script en el diagrama, debe tener sincronizados los parámetros de las anotaciones y los parámetros del código de Apache Spark.

Con el editor de scripts, puede añadir plantillas de código cuando el cursor esté sobre el script. En la parte superior del editor, seleccione las opciones que correspondan:

- Para agregar una tabla de orígenes al script, seleccione Source (Origen).
- Para agregar una tabla de destinos al script, seleccione Target (Destino).
- Para agregar una ubicación de destino al script, seleccione Target location (Ubicación de destino).
- Para agregar una transformación al script, seleccione Transform (Transformación). Para obtener información sobre las funciones que se invocan en el script, consulte Programe scripts AWS Glue ETL en PySpark.
- Para agregar una transformación de espiga al script, seleccione Spigot (Espiga).

En el código insertado, modifique `parameters` en las anotaciones y el código de Apache Spark. Por ejemplo, si añade una transformación Spigot (Espiga), compruebe que `path` se sustituirá tanto en la anotación `@args` como en la línea de código `output`.

La pestaña Logs (Registros) muestra los registros que están asociados con el flujo de trabajo mientras se ejecuta. Aparecen las últimas 1000 líneas.

La pestaña Schema (Esquema) muestra el esquema de los orígenes y los destinos seleccionados, cuando están disponibles en el Data Catalog.

Seguimiento de los datos procesados mediante marcadores de trabajo

AWS Glue realiza un seguimiento de los datos que ya se han procesado durante una ejecución anterior de un trabajo de ETL mediante información de estado persistente de la ejecución del trabajo.

Esta información de estado persistente se llama marcador de flujo de trabajo. Los marcadores de trabajo ayudan a AWS Glue a mantener la información de estado y evitar el reprocesamiento de los datos antiguos. Con los marcadores de trabajo, puede procesar datos nuevos al volver a realizar una ejecución en un intervalo programado. Un marcador de trabajo se compone de los estados de diversos elementos de trabajos, como orígenes, transformaciones y destinos. Por ejemplo, su trabajo de ETL podría leer nuevas particiones en un archivo de Amazon S3. AWS Glue hace el seguimiento de las particiones que el trabajo ha procesado correctamente para evitar procesamiento y datos duplicados en el almacén de datos de destino del trabajo.

Los marcadores de trabajo se implementan para los orígenes de datos JDBC, la transformación Relationalize (Establecimiento de relaciones) y algunos orígenes Amazon Simple Storage Service (Amazon S3). En la siguiente tabla se muestran los formatos de origen Amazon S3 que AWS Glue soporta para los marcadores del trabajo.

Versión de AWS Glue	Formatos de origen de Amazon S3
Versión 0.9	JSON, CSV, Apache Avro, XML
Versión 1.0 y posteriores.	JSON, CSV, Apache Avro, XML, Parquet, ORC

Para obtener información acerca de las versiones de AWS Glue, consulte Definición de propiedades de trabajo para trabajos de Spark.

La característica de marcadores de trabajo tiene funcionalidades adicionales cuando se accede a ella a través de scripts AWS Glue. Al navegar por el script generado, es posible que vea contextos de transformación relacionados con esta característica. Para obtener más información, consulte the section called “Uso de marcadores de trabajo”.

Temas

- Uso de marcadores de trabajo en AWS Glue
- Detalles operativos de la característica de marcadores de trabajo

Uso de marcadores de trabajo en AWS Glue

La opción de marcador de trabajo se transfiere como parámetro al iniciarse el trabajo. En la siguiente tabla se describen las opciones para la configuración de marcadores de trabajo en la consola de AWS Glue.

Job bookmark (Marcador de flujo de trabajo)	Descripción
Habilitado	Hace que el trabajo actualice el estado después de una ejecución para realizar un seguimiento de datos procesados anteriormente. Si el trabajo tiene un origen con compatibilidad de marcador de trabajo, realizará un seguimiento de los datos procesados y cuando se ejecute un trabajo, procesará datos nuevos desde el último punto de control.
Deshabilitado	Los marcadores de trabajo no se utilizan y el trabajo siempre procesa todo el conjunto de datos. Usted es responsable de administrar el resultado de las ejecuciones de trabajos anteriores. Esta es la opción predeterminada.
Pause	Procesar los datos incrementales desde la última ejecución correcta o los datos en el rango identificado por las siguientes subopciones, sin actualizar el estado del último marcador. Usted es responsable de administrar el resultado de las ejecuciones de trabajos anteriores. Las dos subopciones son: • job-bookmark-from <from-value> es el ID de ejecución que represent a toda la entrada que se ha procesado hasta la última ejecución correcta antes de la misma e incluye el ID de ejecución especificado. La entrada correspondiente se ignora. • job-bookmark-to <to-value> es el ID de ejecución que representa toda la entrada que se ha procesado hasta la última ejecución correcta antes de la misma e incluye el ID de ejecución especificado. El trabajo procesa la entrada correspondiente, excluida la entrada identificada por <from-value>. Cualquier entrada posterior a esta entrada también se excluye para el procesamiento. El estado del marcador de trabajo no se actualiza cuando se especifica este conjunto de opciones. Las subopciones son opcionales; sin embargo, cuando se utilizan ambas opciones, se deben proporcionar.

Para obtener información detallada sobre los parámetros transferidos a un trabajo en la línea de comandos y para los marcadores de trabajo específicamente, consulte Parámetros de los trabajos de AWS Glue.

Para orígenes de entrada de Amazon S3, los marcadores de trabajo de AWS Glue comprueban la hora de última modificación de los objetos para verificar qué objetos deben volver a procesarse. Si los datos de origen de entrada se han modificado desde la última ejecución de trabajo, los archivos se vuelven a procesar al ejecutar el trabajo de nuevo.

Para los orígenes JDBC, se aplican las reglas siguientes:

- Para cada tabla, AWS Glue utiliza una o más columnas como claves de marcador para determinar los datos nuevos y procesados. Las claves de marcador se combinan para formar una única clave compuesta.
- AWS Glue utiliza la clave principal como clave de marcador de forma predeterminada, siempre que aumente o disminuya secuencialmente (sin espacios).
- Puede especificar las columnas que se van a utilizar como claves de marcador en el script de AWS Glue. Para obtener más información sobre el uso de marcadores de trabajo en scripts de AWS Glue, consulte the section called “Uso de marcadores de trabajo”.
- AWS Glue no admite el uso de columnas con nombres que distinguen mayúsculas de minúsculas como claves de marcadores de trabajo.

Puede retroceder sus marcadores de trabajo para sus trabajos de Spark ETL de AWS Glue a cualquier ejecución de trabajo anterior. Puede admitir escenarios de reposición de datos mejor al rebobinar sus marcadores de trabajo a cualquier ejecución de trabajo anterior reprocesando los datos solo desde la ejecución del trabajo marcado.

Si va a volver a procesar todos los datos utilizando el mismo trabajo, restablezca el marcador de trabajo. Para restablecer el estado del marcador de trabajo, utilice la consola de AWS Glue, la ResetJobBookmark acción (Python: reset_job_bookmark)operación de API o AWS CLI. Por ejemplo, escriba el siguiente comando usando AWS CLI:

```
aws glue reset-job-bookmark --job-name my-job-name
```

Al retroceder o restablecer un marcador, AWS Glue no limpia los archivos de destino porque podría haber varios destinos y los destinos no se rastrean con marcadores de trabajo. Sólo los archivos

de origen se rastrean con marcadores de trabajo. Puede crear diferentes destinos de salida al retroceder y reprocesar los archivos de origen para evitar datos duplicados en la salida.

AWS Glue realiza el seguimiento de los marcadores de flujo de trabajo para cada flujo de trabajo. Si elimina un flujo de trabajo, el marcador de flujo de trabajo se elimina.

En algunos casos, es posible que tenga habilitados marcadores de trabajo de AWS Glue, pero su trabajo de ETL está reprocesando los datos que ya se han procesado anteriormente. Para obtener información acerca de la resolución de causas comunes de este error, consulte Solución de errores en AWS Glue Spark.

Detalles operativos de la característica de marcadores de trabajo

En esta sección se describen más detalles operativos de utilizar marcadores de trabajo.

Los marcadores de trabajo almacenan los estados de un trabajo. Cada instancia del estado está marcada con un nombre de trabajo y un número de versión. Cuando un script invoca `job.init`, recupera su estado y siempre obtiene la última versión. Dentro de un estado, hay varios elementos de estado, que son específicos de cada origen, transformación e instancia de receptor en el script. Estos elementos de estado se identifican mediante un contexto de transformación que se adjunta al elemento correspondiente (origen, transformación o receptor) en el script. Los elementos de estado se guardan de manera granular cuando `job.commit` se haya invocado desde el script de usuario. El script obtiene el nombre del trabajo y la opción de control de los marcadores de trabajo de los argumentos.

Los elementos de estado del marcador de trabajo son datos específicos de origen, transformación o receptor. Por ejemplo, supongamos que desea leer los datos progresivos desde una ubicación de Amazon S3 a la que un proceso o trabajo ascendente escribe en forma constante. En este caso, el script debe determinar qué se ha procesado hasta el momento. La implementación de marcador de trabajo para el origen de Amazon S3 guarda la información para que cuando el trabajo se ejecute de nuevo, pueda filtrar solo los nuevos objetos mediante la información guardada y recalcular el estado para la próxima ejecución del trabajo. Se utiliza una marca temporal para filtrar los archivos nuevos.

Además de los elementos de estado, los marcadores de trabajo tienen un número de ejecución, un número de intentos y un número de versión. El número de ejecución realiza un seguimiento de la ejecución del trabajo y el número de intentos registra los intentos de ejecución de un trabajo. El número de ejecuciones de un trabajo es un número que se incrementa de forma monotónica por cada ejecución correcta. El número de intentos realiza un seguimiento de los intentos de cada ejecución y solo se incrementa cuando hay una ejecución después de un intento fallido. El número

de versión aumenta de forma monotónica y realiza un seguimiento de las actualizaciones de un marcador de trabajo.

En la base de datos del servicio AWS Glue, los estados de favorito de todas las transformaciones se almacenan juntos como pares clave-valor:

```
{
  "job_name" : ...,
  "run_id": ...,
  "run_number": ..,
  "attempt_number": ...
  "states": {
    "transformation_ctx1" : {
      bookmark_state1
    },
    "transformation_ctx2" : {
      bookmark_state2
    }
  }
}
```

Prácticas recomendadas

A continuación, se indican las prácticas recomendadas para utilizar marcadores de trabajo.

- No cambie la propiedad del origen de datos con el favorito habilitado. Por ejemplo, hay un origen de datos datasource0 que apunta a una ruta de entrada A de Simple Storage Service (Amazon S3), y el trabajo ha estado leyendo desde un origen que se ha estado ejecutando durante varias rondas con el favorito habilitado. Si cambia la ruta de entrada de datasource0 a la ruta B de Simple Storage Service (Amazon S3) sin cambiar `transformation_ctx`, el trabajo de AWS Glue utilizará el antiguo estado de favorito almacenado. Eso provocará que se omitan o falten archivos en la ruta de entrada B, ya que AWS Glue presupondrá que esos archivos se han procesado en ejecuciones anteriores.
- Utilice una tabla de catálogo con favoritos para mejorar la administración de particiones. Los favoritos funcionan para orígenes de datos de Data Catalog o de opciones. No obstante, resulta difícil eliminar/agregar nuevas particiones con el enfoque de opciones. Utilizar una tabla de catálogo con rastreadores puede proporcionar una mejor automatización para realizar un seguimiento de las nuevas particiones agregadas, y ofrece flexibilidad para seleccionar particiones concretas con un predicado de inserción.

- Utilice el enumerador de archivos de Simple Storage Service (Amazon S3) de AWS Glue para conjuntos de datos de gran tamaño. Un favorito mostrará una lista de todos los archivos de cada partición de entrada y realizará el filtrado, de modo que, si hay demasiados archivos en una sola partición, puede que el controlador del favorito se quede sin suficiente memoria. Utilice el enumerador de archivos de Simple Storage Service (Amazon S3) de AWS Glue para evitar mostrar una lista de todos los archivos de la memoria a la vez.

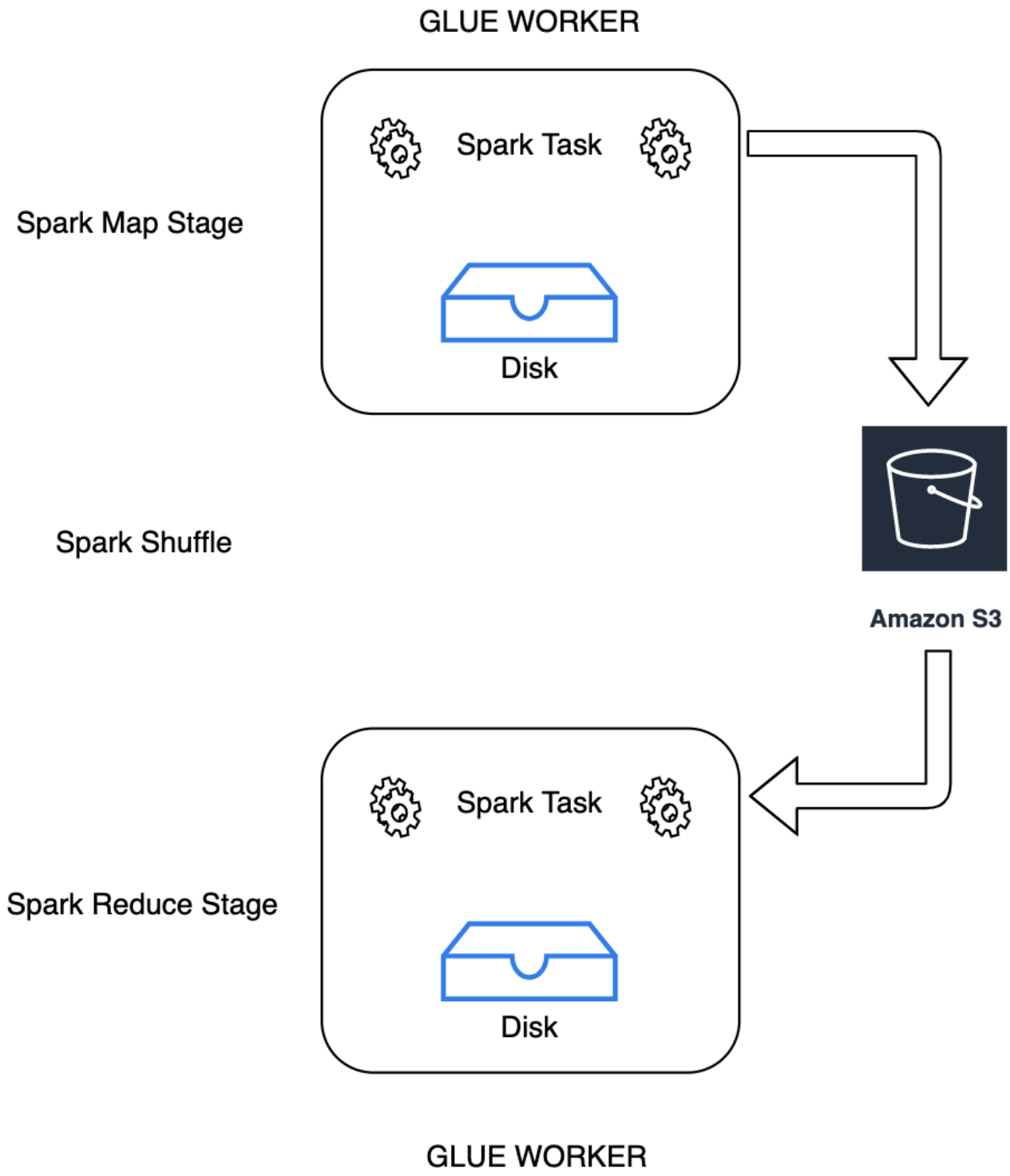
Complemento de mezclas aleatorias de AWS Glue Spark con Amazon S3

El mezclado aleatorio es un paso importante en un trabajo de Spark siempre que los datos se reorganizan entre particiones. Esto es necesario porque transformaciones de gran extensión como `join`, `groupByKey`, `reduceByKey`, y `repartition` requieren información de otras particiones para completar el procesamiento. Spark recopila los datos requeridos de cada partición y los combina en una nueva partición. Durante una mezcla aleatoria, los datos se escriben en el disco y se transfieren a través de la red. Como resultado, la operación de mezcla aleatoria está vinculada a la capacidad del disco local. Spark lanza un error `No space left on device` o `MetadataFetchFailedException` cuando no queda suficiente espacio en disco en el ejecutor y no se puede realizar la recuperación.

> **Note**
>
> El complemento AWS Glue Spark shuffle de Amazon S3 solo es compatible con los trabajos de ETL AWS Glue.

Solución

Con AWS Glue, ahora puede utilizar Amazon S3 para almacenar datos de mezclas aleatorias de Spark. Amazon Simple Storage Service (Amazon S3) es un servicio de almacenamiento de objetos que ofrece escalabilidad, disponibilidad de datos, seguridad y rendimiento líderes del sector. Esta solución separa la informática y el almacenamiento para sus trabajos de Spark y proporciona una elasticidad completa y un almacenamiento de mezclas aleatorias de bajo costo, lo que le permite ejecutar con confianza sus cargas de trabajo con mayor cantidad de mezclas aleatorias.



Presentamos Cloud Shuffle Storage, un nuevo complemento para Apache Spark para usar Amazon S3. Puede habilitar la función de mezclado aleatorio de Amazon S3 para ejecutar sus trabajos de AWS Glue con confianza y sin errores si se sabe que están vinculados por la capacidad del disco local para operaciones de mezclas aleatorias grandes. En algunos casos, el mezclado aleatorio

en Amazon S3 es ligeramente más lento que el disco local (o EBS) si tiene un gran número de pequeñas particiones o archivos de mezcla aleatoria escritos en Amazon S3.

Requisitos previos para usar el complemento de almacenamiento de Cloud Shuffle

Para usar el complemento de almacenamiento de Cloud Shuffle con trabajos de AWS Glue ETL, necesita lo siguiente:

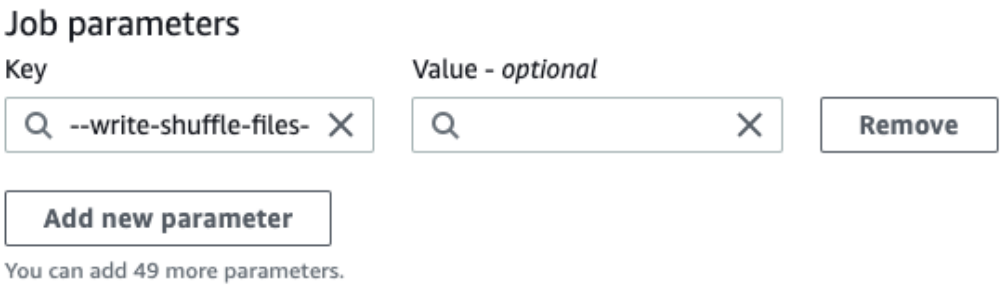
- Un bucket de Amazon S3 ubicado en la misma región en la que se ejecutó el trabajo, para almacenar los datos intermedios mezclados y volcados. El prefijo Amazon S3 del almacenamiento aleatorio se puede especificar con `--conf spark.shuffle.glue.s3ShuffleBucket=s3://`*`shuffle-bucket`*`/`*`prefix`*`/`, como en el siguiente ejemplo:

  ```
  --conf spark.shuffle.glue.s3ShuffleBucket=s3://glue-shuffle-123456789-us-east-1/glue-shuffle-data/
  ```

- Establezca las políticas de ciclo de vida de almacenamiento de Amazon S3 en el prefijo (como `glue-shuffle-data`) ya que el administrador de mezclas aleatorias no limpia los archivos una vez finalizado el trabajo. Los datos intermedios mezclados y volcados se deben eliminar una vez finalizado el trabajo. Los usuarios pueden establecer políticas de ciclo de vida breves en el prefijo. Las instrucciones para configurar la política de ciclo de vida de Amazon S3 están disponibles en Configurar el ciclo de vida de un bucket en la Guía del usuario de Amazon Simple Storage Service.

Uso del administrador de mezclas aleatorias de AWS Glue Spark desde la consola de AWS

Para configurar el administrador de mezclas aleatorias de AWS Glue Spark con la consola de AWS Glue o AWS Glue Studio cuando configura un trabajo: elija el parámetro de trabajo: --write-shuffle-files-to-s3 para habilitar el mezclado aleatorio de Amazon S3 para el trabajo.



Uso del complemento de mezclas aleatorias de AWS Glue Spark

Los siguientes parámetros de trabajo activan y establecen los ajustes del administrador de mezclas aleatorias de AWS Glue. Estos parámetros son indicadores, por lo que no se tienen en cuenta los valores proporcionados.

- `--write-shuffle-files-to-s3`: el valor de la marca principal habilita la opción del administrador de mezclas aleatorias de AWS Glue Spark para utilizar buckets de Amazon S3 para escribir y leer datos aleatorios. Cuando la marca no se especifica, no se utiliza el administrador de mezclas aleatorias.
- `--write-shuffle-spills-to-s3`: (solo se admite en la versión 2.0 de AWS Glue). Una marca opcional que permite descargar archivos de volcado en buckets de Amazon S3, lo que proporciona resistencia adicional al trabajo de Spark. Esto solo es necesario para cargas de trabajo grandes que provocan grandes desbordamientos al disco. Cuando no se especifica la marca, no se graba ningún archivo de volcado intermedio.
- `--conf spark.shuffle.glue.s3ShuffleBucket=s3://<shuffle-bucket>`: otra marca opcional que especifica el bucket de Amazon S3 en el que se escriben los archivos de mezcla aleatoria. De forma predeterminada, `--TempDir`/shuffle-data. AWS Glue 3.0+ permite escribir archivos aleatorios en varios buckets al especificar los buckets con un delimitador de comas, como en `--conf spark.shuffle.glue.s3ShuffleBucket=s3://`*`shuffle-bucket-1`*`/`*`prefix`*`,s3://`*`shuffle-bucket-2`*`/`*`prefix`*`/`. El uso de varios buckets mejora el rendimiento.

Debe proporcionar los ajustes de configuración de seguridad para habilitar el cifrado en reposo de los datos aleatorios. Para obtener más información sobre las configuraciones de seguridad, consulte the section called “Configuración del cifrado”. AWS Glue admite todas las demás configuraciones relacionadas con mezclas aleatorias proporcionadas por Spark.

Archivos binarios del software para el complemento Cloud Shuffle Storage

También puede descargar los archivos binarios del software del complemento Cloud Shuffle Storage para Apache Spark bajo la licencia Apache 2.0 y ejecutarlo en cualquier entorno de Spark. El nuevo complemento incluye compatibilidad lista para usar con Amazon S3 y también se puede configurar fácilmente para usar otras formas de almacenamiento en la nube, como Google Cloud Storage y Microsoft Azure Blob Storage. Para obtener más información, consulte Cloud Shuffle Storage Plugin for Apache Spark (Complemento Cloud Shuffle Storage para Apache Spark).

Notas y limitaciones

Las siguientes son notas o limitaciones para el administrador de mezclas aleatorias de AWS Glue:

- El administrador de mezclas aleatorias AWS Glue no elimina automáticamente los archivos de datos de mezcla aleatoria (temporales) almacenados en un bucket de Amazon S3 después de completar un trabajo. Para garantizar la protección de los datos, siga las instrucciones que se indican en Requisitos previos para usar el complemento de almacenamiento de Cloud Shuffle antes de activar el complemento Cloud Shuffle Storage Plugin.
- Puede utilizar esta característica si los datos están sesgados.

Complemento Cloud Shuffle Storage para Apache Spark

El complemento Cloud Shuffle Storage es un complemento de Apache Spark compatible con la API de `ShuffleDataIO` que permite almacenar datos aleatorios en sistemas de almacenamiento en la nube (como Amazon S3). Ayuda a complementar o reemplazar la capacidad de almacenamiento del disco local para operaciones aleatorias de gran tamaño que suelen desencadenarse por transformaciones como `join`, `reduceByKey`, `groupByKey` y `repartition` en las aplicaciones de Spark, lo que reduce los errores comunes o la dislocación de precio y rendimiento de las canalizaciones y los trabajos de análisis de datos sin servidor.

AWS Glue

Las versiones 3.0 y 4.0 de AWS Glue incluyen el complemento preinstalado y listo para permitir la transferencia aleatoria a Amazon S3 sin ningún paso adicional. Para más información, consulte Complemento de mezclas aleatorias de Spark para AWS Glue con Amazon S3 para habilitar la característica en las aplicaciones de Spark.

Otros entornos de Spark

El complemento requiere que se establezcan las siguientes configuraciones de Spark en otros entornos de Spark:

- `--conf spark.shuffle.sort.io.plugin.class=com.amazonaws.spark.shuffle.io.cloud.Choppe` esto indica a Spark que debe utilizar este complemento para Shuffle IO.
- `--conf spark.shuffle.storage.path=s3://`*`bucket-name`*`/`*`shuffle-file-dir`*: la ruta en la que se almacenarán los archivos aleatorios.

> **Note**
>
> El complemento sobrescribe una clase principal de Spark. Como resultado, el contenedor del complemento debe cargarse antes que los contenedores de Spark. Para ello, utilice `userClassPathFirst` en entornos YARN locales si el complemento se utiliza fuera de AWS Glue.

Agrupación del complemento con las aplicaciones de Spark

Puede combinar el complemento con las aplicaciones de Spark y distribuciones de Spark (versiones 3.1 y posteriores) si agrega la dependencia del complemento en `pom.xml` de Maven mientras desarrolla las aplicaciones de Spark de manera local. Para más información acerca del complemento y las versiones de Spark, consulte Versiones del complemento.

```
<repositories>
   ...
    <repository>
        <id>aws-glue-etl-artifacts</id>
        <url>https://aws-glue-etl-artifacts.s3.amazonaws.com/release/ </url>
    </repository>
</repositories>
...
<dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>chopper-plugin</artifactId>
    <version>3.1-amzn-LATEST</version>
</dependency>
```

También puede descargar los archivos binarios de los artefactos de Maven de AWS Glue directamente e incluirlos en la aplicación de Spark de la siguiente manera.

```
#!/bin/bash
sudo wget -v https://aws-glue-etl-artifacts.s3.amazonaws.com/release/com/amazonaws/
chopper-plugin/3.1-amzn-LATEST/chopper-plugin-3.1-amzn-LATEST.jar -P /usr/lib/spark/
jars/
```

Ejemplo spark-submit

```
spark-submit --deploy-mode cluster \
--conf spark.shuffle.storage.s3.path=s3://<ShuffleBucket>/<shuffle-dir> \
```

```
--conf spark.driver.extraClassPath=<Path to plugin jar> \
--conf spark.executor.extraClassPath=<Path to plugin jar> \
--class <your test class name> s3://<ShuffleBucket>/<Your application jar> \
```

Opciones de configuración opcionales

Estos son los valores de las opciones de configuración opcionales que controlan el comportamiento aleatorio de Amazon S3.

- `spark.shuffle.storage.s3.enableServerSideEncryption`: habilita y desactiva SSE de S3 para ordenar aleatoriamente y volcar archivos. El valor predeterminado es `true`.
- `spark.shuffle.storage.s3.serverSideEncryption.algorithm`: el algoritmo de SSE que se va a usar. El valor predeterminado es `AES256`.
- `spark.shuffle.storage.s3.serverSideEncryption.kms.key`: el ARN de la clave de KMS cuando aws:kms de SSE está activado.

Junto con estas configuraciones, es posible que deba establecer configuraciones como`spark.hadoop.fs.s3.enableServerSideEncryption`, y otras configuraciones específicas del entorno para garantizar que se aplique el cifrado adecuado para su caso de uso.

Versiones del complemento

Este complemento es compatible con las versiones de Spark asociadas a cada versión de AWS Glue. En la siguiente tabla, se muestra la versión de AWS Glue, la versión de Spark y la versión del complemento asociada a la ubicación de Amazon S3 del binario de software del complemento.

Versión de AWS Glue	Versión de Spark	Versión del complemento	Ubicación de Amazon S3
3.0	3.1	3.1-amzn-LATEST	s3://aws-glue-etl-artifacts/release/com/amazonaws/chopper-plugin/3.1-amzn-0/chopper-plugin-3.1-amzn-LATEST.jar
4.0	3.3	3.3-amzn-LATEST	s3://aws-glue-etl-artifacts/release/com/

Versión de AWS Glue	Versión de Spark	Versión del complemento	Ubicación de Amazon S3
			amazonaws/chopper-plugin/3.3-amzn-0/chopper-plugin-3.3-amzn-LATEST.jar

Licencia

El binario de software de este complemento cuenta con la licencia de Apache-2.0.

Supervisión de trabajos de Spark de AWS Glue

Temas

- Spark Metrics disponible en AWS Glue Studio
- Monitorización de trabajos mediante la interfaz de usuario web de Apache Spark
- Supervisión con Información de ejecuciones de trabajos de AWS Glue
- Monitoreo de con Amazon CloudWatch
- Monitorización y depuración de trabajo

Spark Metrics disponible en AWS Glue Studio

La pestaña Metrics (Métricas) muestra las métricas recopiladas cuando se ejecuta un trabajo y se habilita la creación de perfiles. En los trabajos de Spark se muestran los gráficos siguientes:

- Movimiento de datos de ETL
- Perfil de la memoria: controlador y ejecutores

Elija View additional metrics (Ver métricas adicionales) para mostrar los siguientes gráficos:

- Movimiento de datos de ETL
- Perfil de la memoria: controlador y ejecutores
- Mezcla de datos entre los ejecutores
- Carga de la CPU: controlador y ejecutores

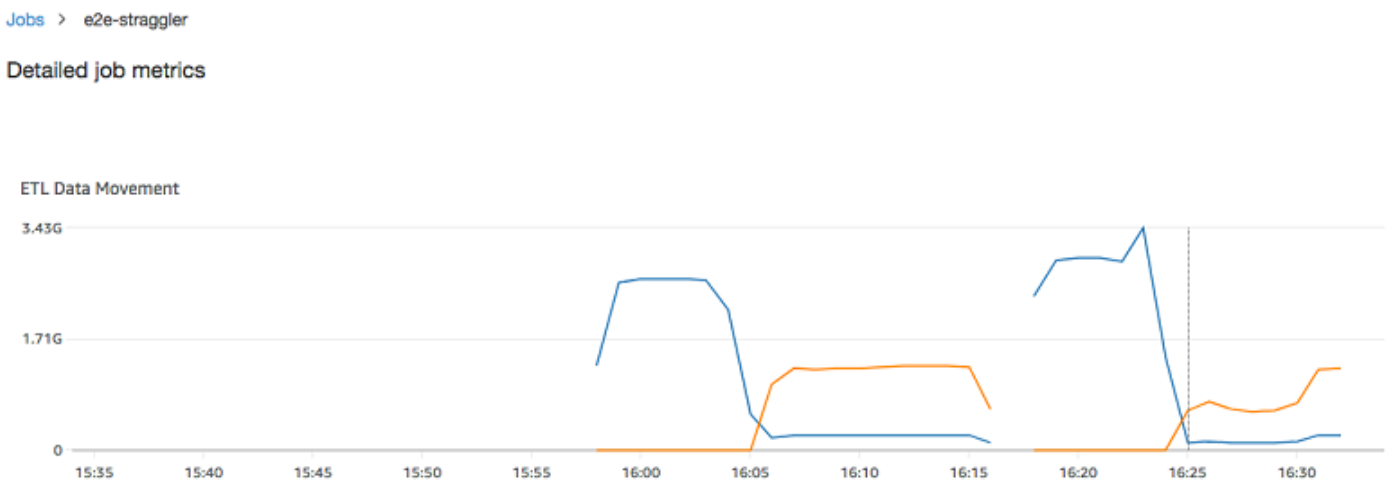
- Ejecución de trabajo: ejecutores activos, etapas completadas y número máximo de ejecutores necesarios

Los datos de estos gráficos se envían a CloudWatch las métricas si el trabajo está configurado para recopilar métricas. Para obtener más información acerca de cómo habilitar métricas e interpretar los gráficos, consulte Monitorización y depuración de trabajo.

Example Gráfico de movimiento de datos de ETL

En el gráfico de movimiento de datos de ETL se muestran las siguientes métricas:

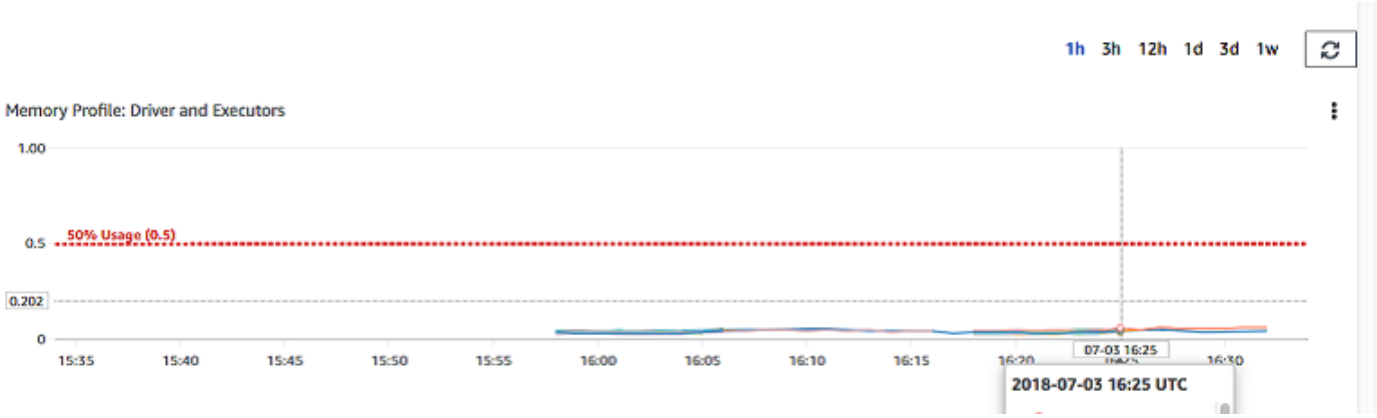
- El número de bytes que leen todos los ejecutores de Amazon S3 —`glue.ALL.s3.filesystem.read_bytes`
- El número de bytes que escriben todos los ejecutores en Amazon S3 —`glue.ALL.s3.filesystem.write_bytes`



Example Gráfico de perfil de la memoria

En el gráfico de perfil de la memoria se muestran las siguientes métricas:

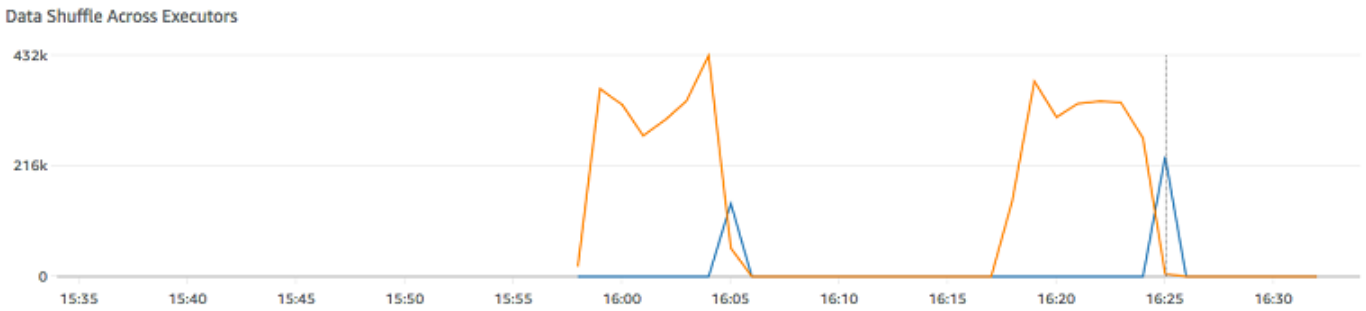
- La fracción de la memoria usada por el montón de JVM para este controlador (escala: de 0 a 1), un ejecutor identificado por executorId, o todos los ejecutores.
  - `glue.driver.jvm.heap.usage`
  - `glue.executorId.jvm.heap.usage`
  - `glue.ALL.jvm.heap.usage`



Example Gráfico de mezcla de datos entre los ejecutores

En el gráfico Mezcla de datos entre los ejecutores se muestran las siguientes métricas:

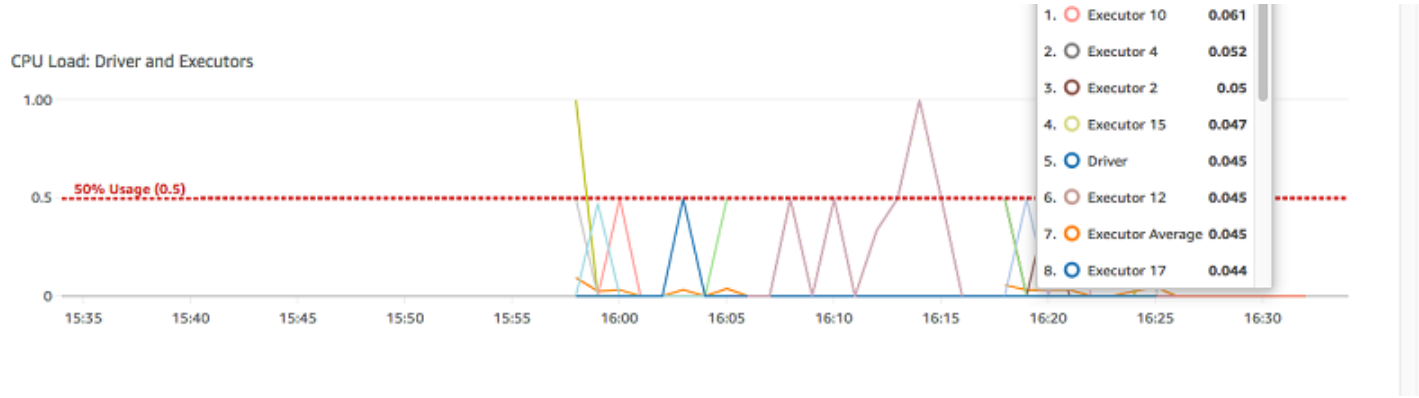
- El número de bytes que leen todos los ejecutores para mezclar los datos entre ellos: `glue.driver.aggregate.shuffleLocalBytesRead`
- El número de bytes que escriben todos los ejecutores para mezclar los datos entre ellos: `glue.driver.aggregate.shuffleBytesWritten`



Example Gráfico de carga de la CPU

En el gráfico de carga de la CPU se muestran las siguientes métricas:

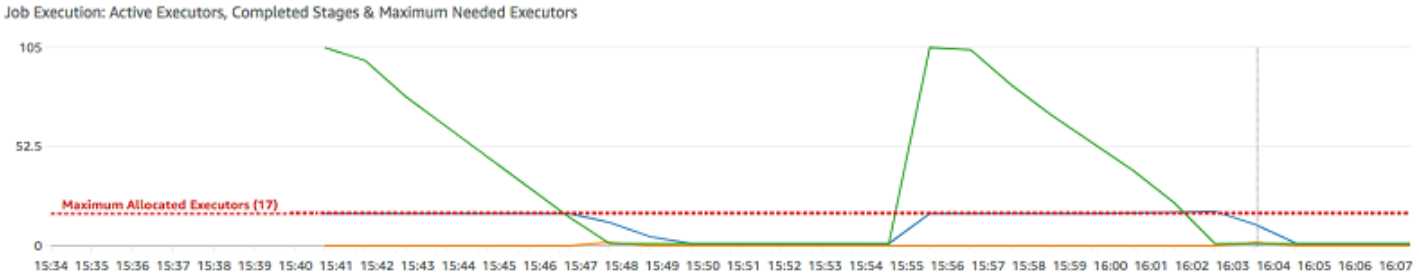
- La fracción de la carga del sistema de CPU usada (escala: de 0 a 1) por el controlador, un ejecutor identificado por executorId, o todos los ejecutores.
  - `glue.driver.system.cpuSystemLoad`
  - `glue.executorId.system.cpuSystemLoad`
  - `glue.ALL.system.cpuSystemLoad`



Example Gráfico de ejecución de trabajo

En el gráfico de ejecución de trabajo se muestran las siguientes métricas:

- El número de ejecutores que se ejecutan activamente: `glue.driver.ExecutorAllocationManager.executors.numberAllExecutors`
- El número de etapas completadas: `glue.aggregate.numCompletedStages`
- El número máximo de ejecutores necesarios: `glue.driver.ExecutorAllocationManager.executors.numberMaxNeededExecutors`



Monitorización de trabajos mediante la interfaz de usuario web de Apache Spark

Puede utilizar la interfaz de usuario web de Apache Spark para monitorizar y depurar trabajos de ETL AWS Glue que se ejecutan en el sistema de trabajos AWS Glue, así como aplicaciones de Spark que se ejecutan en puntos de enlace de desarrollo AWS Glue. La interfaz de usuario de Spark le permite comprobar lo siguiente para cada trabajo:

- La escala de tiempo del evento de cada etapa de Spark
- Un gráfico acíclico dirigido (DAG) del trabajo
- Planes físicos y lógicos para consultas de SparkSQL

- Las variables de entorno de Spark subyacentes para cada trabajo

Para obtener más información sobre el uso de la interfaz de usuario web de Spark, consulta la Interfaz de usuario web en la documentación de Spark. Para obtener orientación sobre cómo interpretar los resultados de la interfaz de usuario de Spark para mejorar el rendimiento de tu trabajo, consulta las prácticas recomendadas AWS Glue para ajustar el rendimiento de los trabajos de Apache Spark en la Guía AWS prescriptiva.

Puedes ver la interfaz de usuario de Spark en la AWS Glue consola. Está disponible cuando un AWS Glue trabajo se ejecuta en la versión AWS Glue 3.0 o versiones posteriores y los registros se generan en el formato estándar (en lugar de heredado), que es el predeterminado para los trabajos más recientes. Si tiene archivos de registro de más de 0,5 GB, puede habilitar la compatibilidad con el registro continuo para las ejecuciones de trabajos en versiones AWS Glue 4.0 o posteriores para simplificar el archivado de registros, el análisis y la solución de problemas.

Puedes activar la interfaz de usuario de Spark desde la AWS Glue consola o desde AWS Command Line Interface (AWS CLI). Al habilitar la interfaz de usuario de Spark, los trabajos de ETL de AWS Glue y las aplicaciones de Spark en puntos de conexión de desarrollo de AWS Glue pueden hacer una copia de seguridad de los registros de eventos de Spark en una ubicación que especifique en Amazon Simple Storage Service (Amazon S3). Los registros de eventos respaldados en Amazon S3 se pueden utilizar con la interfaz de usuario de Spark tanto en tiempo real mientras se está ejecutando el trabajo como después de que se haya completado el trabajo. Mientras los registros permanezcan en Amazon S3, la interfaz de usuario de Spark de la AWS Glue consola podrá verlos.

Permisos

Para usar la interfaz de usuario de Spark en la AWS Glue consola, puedes usar `UseGlueStudio` o añadir todas las API de servicio individuales. Para utilizar la interfaz de Spark, todas las API son necesarias; sin embargo, los usuarios pueden acceder a las características de la interfaz de Spark al agregar sus API de servicio en el permiso de IAM para un acceso detallado.

`RequestLogParsing` es la más importante, ya que realiza el análisis de los registros. Las API restantes realizan la lectura de los respectivos datos analizados. Por ejemplo, `GetStages` brinda acceso a los datos sobre las etapas de un trabajo de Spark.

La lista de las API de servicio de la interfaz de usuario de Spark elaboradas para `UseGlueStudio` se encuentra a continuación, en la política de muestra. La política a continuación brinda acceso para que solo se utilicen las características de la interfaz de Spark. Para añadir más permisos, como Amazon S3 e IAM, consulte Creación de políticas de IAM personalizadas para. AWS Glue Studio

La lista de las API de servicio de la interfaz de usuario de Spark elaboradas para `UseGlueStudio` se encuentra a continuación, en la política de muestra. Al utilizar una API de servicio de la interfaz de Spark, utilice el espacio de nombre `glue:<ServiceAPI>`.

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Sid": "AllowGlueStudioSparkUI",
   "Effect": "Allow",
            "Action": [
                "glue:RequestLogParsing",
                "glue:GetLogParsingStatus",
                "glue:GetEnvironment",
                "glue:GetJobs",
                "glue:GetJob",
                "glue:GetStage",
                "glue:GetStages",
                "glue:GetStageFiles",
                "glue:BatchGetStageFiles",
                "glue:GetStageAttempt",
                "glue:GetStageAttemptTaskList",
                "glue:GetStageAttemptTaskSummary",
                "glue:GetExecutors",
                "glue:GetExecutorsThreads",
                "glue:GetStorage",
                "glue:GetStorageUnit",
                "glue:GetQueries",
                "glue:GetQuery"
            ],
   "Resource": [
    "*"
   ]
  }
 ]
}
```

Limitaciones

- La interfaz de usuario de Spark en la AWS Glue consola no está disponible para trabajos ejecutados antes del 20 de noviembre de 2023 porque están en el formato de registro antiguo.

- La interfaz de usuario de Spark en la AWS Glue consola admite los registros sucesivos de la AWS Glue versión 4.0, como los que se generan de forma predeterminada en los trabajos de streaming. La suma máxima de todos los archivos de eventos acumulados generados es de 2 GB. Para los AWS Glue trabajos que no admiten registros acumulados, el tamaño máximo del archivo de eventos de registro que admite SparkUI es de 0,5 GB.
- La interfaz de usuario de Spark sin servidor no está disponible para los registros de eventos de Spark almacenados en un bucket de Amazon S3 al que solo puede acceder tu VPC.

Ejemplo: interfaz de usuario web de Apache Spark

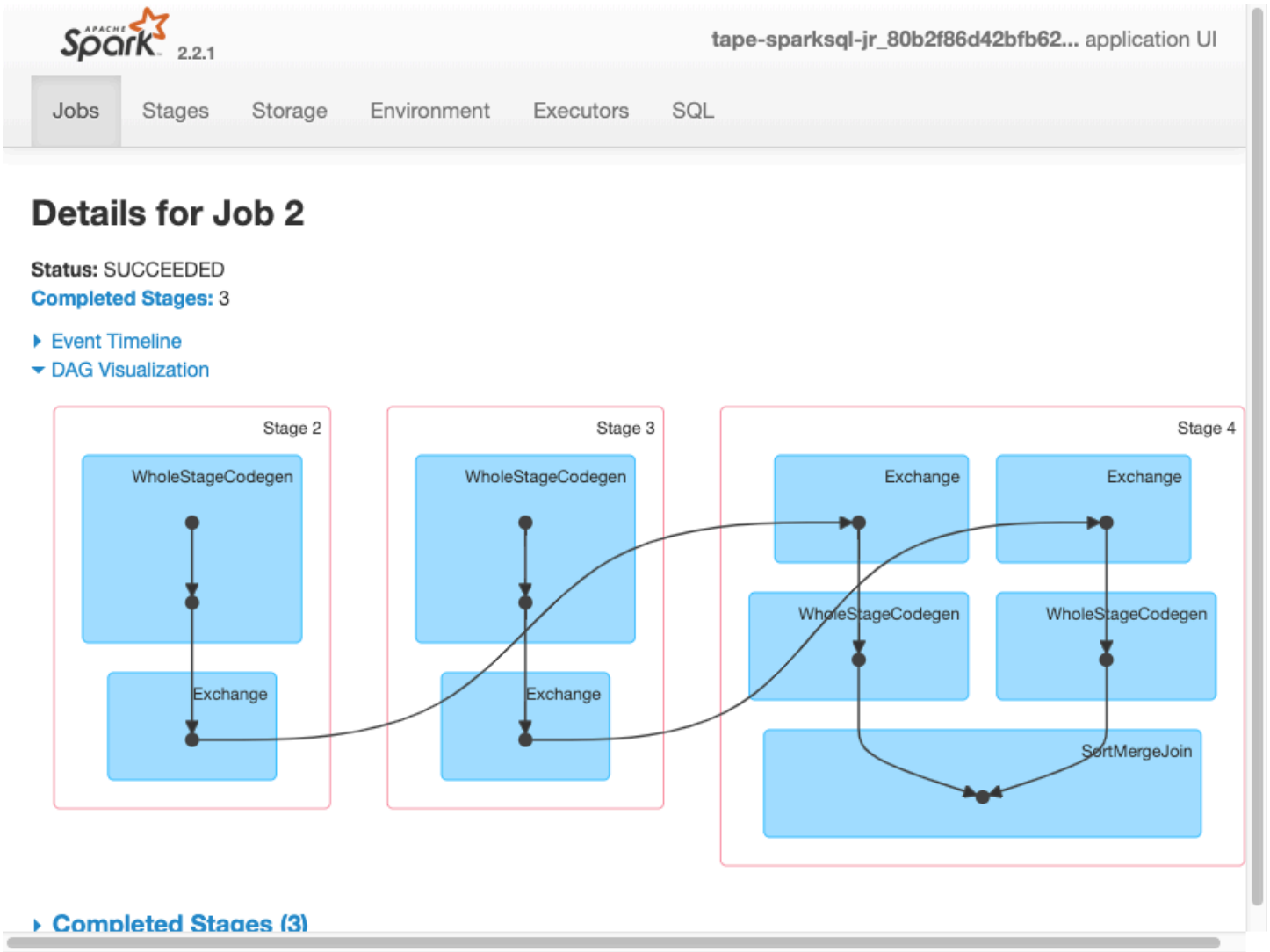
Este ejemplo le muestra cómo utilizar la interfaz de usuario de Spark para entender su desempeño laboral. Las capturas de pantalla muestran la interfaz de usuario web de Spark proporcionada por un servidor de historial de Spark autogestionado. La interfaz de usuario de Spark en la AWS Glue consola ofrece vistas similares. Para obtener más información sobre el uso de la interfaz de usuario web de Spark, consulta la Interfaz de usuario web en la documentación de Spark.

A continuación, se muestra un ejemplo de una aplicación Spark que lee desde dos orígenes de datos, realiza una transformación de combinación y la escribe en Amazon S3 en formato Parquet.

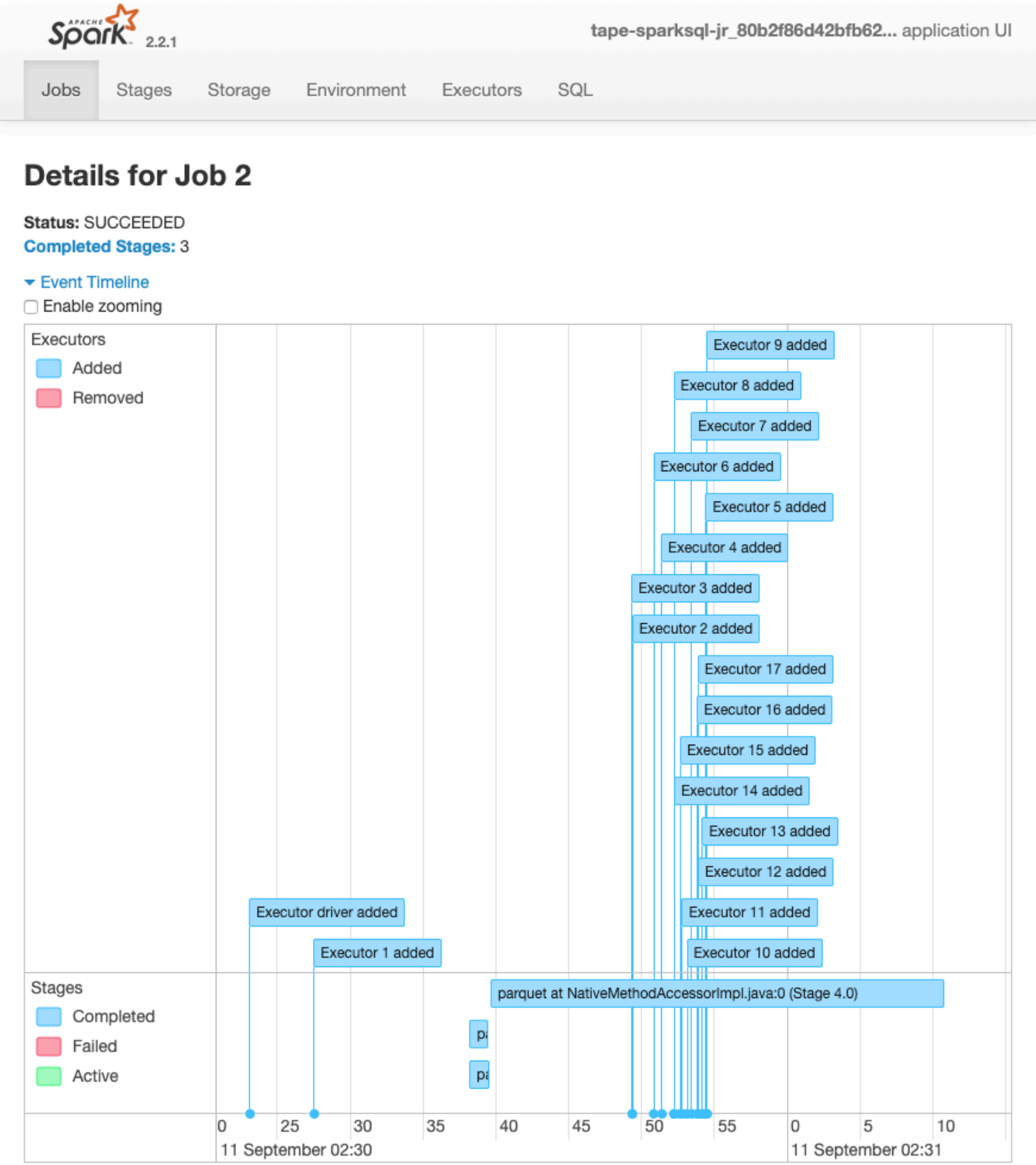
```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job
from pyspark.sql.functions import count, when, expr, col, sum, isnull
from pyspark.sql.functions import countDistinct
from awsglue.dynamicframe import DynamicFrame

args = getResolvedOptions(sys.argv, ['JOB_NAME'])

sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session

job = Job(glueContext)
job.init(args['JOB_NAME'])

df_persons = spark.read.json("s3://awsglue-datasets/examples/us-legislators/all/
persons.json")
```

```
df_memberships = spark.read.json("s3://awsglue-datasets/examples/us-legislators/all/
memberships.json")

df_joined = df_persons.join(df_memberships, df_persons.id == df_memberships.person_id,
 'fullouter')
df_joined.write.parquet("s3://aws-glue-demo-sparkui/output/")

job.commit()
```

La siguiente visualización del DAG muestra las diferentes etapas de este trabajo de Spark.



La siguiente escala de tiempo de eventos para un trabajo muestra el inicio, la ejecución y la terminación de diferentes ejecutores de Spark.


Spark 2.2.1
tape-sparksql-jr_80b2f86d42bfb62... application UI
Jobs
Stages
Storage
Environment
Executors
SQL
Details for Job 2
Status: SUCCEEDED
Completed Stages: 3
Event Timeline
Enable zooming
Executors
Added
Removed
Executor 9 added
Executor 8 added
Executor 7 added
Executor 6 added
Executor 5 added
Executor 4 added
Executor 3 added
Executor 2 added
Executor 17 added
Executor 16 added
Executor 15 added
Executor 14 added
Executor 13 added
Executor 12 added
Executor 11 added
Executor 10 added
Executor driver added
Executor 1 added
Stages
Completed
Failed
Active
parquet at NativeMethodAccessorImpl.java:0 (Stage 4.0)
0
25
30
35
40
45
50
55
0
5
10
11 September 02:30
11 September 02:31
DAG Visualization
Completed Stages (3)

En la siguiente pantalla se muestran los detalles de los planes de consulta de SparkSQL:

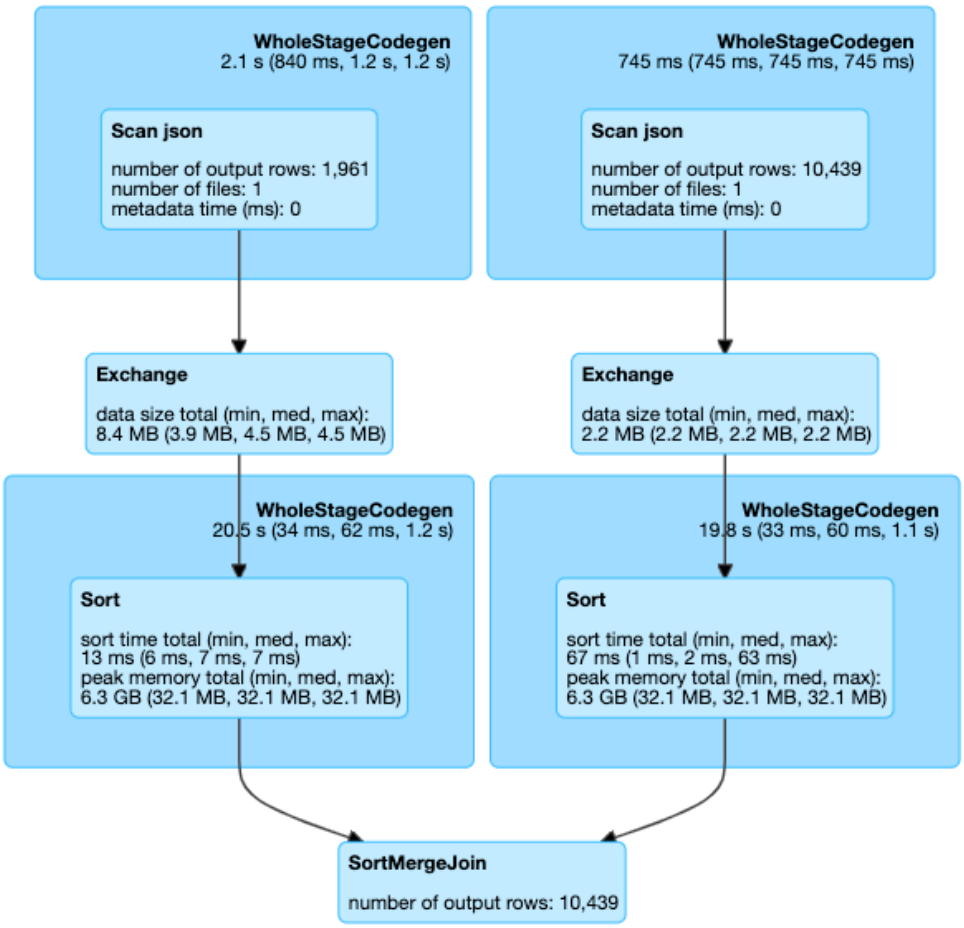
- Plan lógico examinado
- Plan lógico analizado
- Plan lógico optimizado
- Plan físico para la ejecución

Spark 2.2.1 tape-sparksql-jr_80b2f86d42bfb62... application UI

Jobs Stages Storage Environment Executors SQL

Details for Query 0

Submitted Time: 2019/09/11 02:30:37
Duration: 34 s
Succeeded Jobs: 2



▾ Details

```
== Parsed Logical Plan ==
Join FullOuter, (id#14 = person_id#50)
:-
Relation[birth_date#8,contact_details#9,death_date#10,family_name#11,gender#12,given_name#13,id#14,identifiers#15
,image#16,images#17,links#18,name#19,other_names#20,sort_name#21] json
+-
Relation[area_id#45,end_date#46,legislative_period_id#47,on_behalf_of_id#48,organization_id#49,person_id#50,role#
51,start_date#52] json

== Analyzed Logical Plan ==
birth_date: string, contact_details: array<struct<type:string,value:string>>, death_date: string, family_name:
string, gender: string, given_name: string, id: string, identifiers:
array<struct<identifier:string,scheme:string>>, image: string, images: array<struct<url:string>>, links:
array<struct<note:string,url:string>>, name: string, other_names:
array<struct<lang:string,name:string,note:string>>, sort_name: string, area_id: string, end_date: string,
legislative_period_id: string, on_behalf_of_id: string, organization_id: string, person_id: string, role: string,
start_date: string
Join FullOuter, (id#14 = person_id#50)
```

Temas

- Habilitación de la interfaz de usuario web de Apache Spark para trabajos de AWS Glue
- Lanzamiento del servidor de historial de Spark

Habilitación de la interfaz de usuario web de Apache Spark para trabajos de AWS Glue

Puede utilizar la interfaz de usuario web de Apache Spark para monitorizar y depurar trabajos de ETL AWS Glue que se ejecutan en el sistema de trabajos AWS Glue. Puede configurar la interfaz de usuario de Spark mediante la consola de AWS Glue o la AWS Command Line Interface (AWS CLI).

Cada 30 segundos, AWS Glue hace una copia de seguridad de los registros de eventos de Spark a la ruta de Amazon S3 que especifique.

Temas

- Configuración de la interfaz de usuario de Spark (consola)
- Configuración de la interfaz de usuario de Spark (AWS CLI)
- Configuración de la interfaz de usuario de Spark para sesiones con cuadernos
- Habilita los registros continuos

Configuración de la interfaz de usuario de Spark (consola)

Siga estos pasos para configurar la interfaz de usuario de Spark mediante la AWS Management Console. Al crear un AWS Glue trabajo, la interfaz de usuario de Spark está habilitada de forma predeterminada.

Activar la interfaz de Spark al crear o editar un trabajo

1. Inicia sesión AWS Management Console y abre la AWS Glue consola en https://console.aws.amazon.com/glue/.
2. En el panel de navegación, seleccione Trabajos.
3. Elija Agregar trabajo o seleccione uno existente.
4. En Detalles del trabajo, abra las Propiedades avanzadas.
5. En la pestaña Interfaz de usuario de Spark, seleccione Escribir registros de interfaz de usuario de Spark en Amazon S3.
6. Especifique una ruta de Amazon S3 para almacenar los registros de eventos de Spark para el trabajo. Tenga en cuenta que cuando utiliza una configuración de seguridad en el trabajo, el

cifrado también se aplica al archivo de registro de la interfaz de usuario de Spark. Para obtener más información, consulte Cifrado de datos escritos por AWS Glue.

7. En la Configuración de registro y monitoreo de la interfaz de usuario de Spark:

   - Seleccione Estándar si va a generar registros para verlos en la AWS Glue consola.
   - Si está generando registros para verlos en un servidor de historial de Spark, seleccione Heredado.
   - También puede optar por generar ambos.

Configuración de la interfaz de usuario de Spark (AWS CLI)

Para generar registros para verlos con la interfaz de usuario de Spark, en la AWS Glue consola, usa AWS CLI para pasar los siguientes parámetros de trabajo a los AWS Glue trabajos. Para obtener más información, consulte the section called “Parámetros del flujo de trabajo”.

```
'--enable-spark-ui': 'true',
'--spark-event-logs-path': 's3://s3-event-log-path'
```

Para distribuir los registros a sus ubicaciones antiguas, establezca el parámetro `--enable-spark-ui-legacy-path` en `"true"`. Si no desea generar registros en ambos formatos, elimine el parámetro `--enable-spark-ui`.

Configuración de la interfaz de usuario de Spark para sesiones con cuadernos

> **Warning**
>
> AWS Glue Por el momento, las sesiones interactivas no admiten la interfaz de usuario de Spark en la consola. Configure un servidor de historial de Spark.

Si utilizas AWS Glue ordenadores portátiles, configura la configuración de SparkUI antes de iniciar la sesión. Para ello, utilice el comando mágico celular `%%configure`:

```
%%configure { “--enable-spark-ui”: “true”, “--spark-event-logs-path”: “s3://path” }
```

Habilita los registros continuos

Habilitar SparkUI y los archivos de eventos de registro continuo para los AWS Glue trabajos ofrece varias ventajas:

- Archivos de registro continuo de eventos: al habilitar los archivos de registro continuo de eventos, AWS Glue genera archivos de registro independientes para cada paso de la ejecución del trabajo, lo que facilita la identificación y la solución de problemas específicos de una etapa o transformación en particular.
- Mejor administración del registro: los archivos de registro continuo de eventos ayudan a administrar los archivos de registro de manera más eficiente. En lugar de tener un único archivo de registro, potencialmente grande, los registros se dividen en archivos más pequeños y fáciles de administrar en función de las etapas de ejecución del trabajo. Esto puede simplificar el archivado, el análisis y la solución de problemas de los registros.
- Tolerancia a errores mejorada: si un AWS Glue trabajo falla o se interrumpe, los archivos de registro continuo de eventos pueden proporcionar información valiosa sobre la última etapa exitosa, lo que facilita reanudar el trabajo desde ese punto en lugar de empezar desde cero.
- Optimización de costos: al habilitar los archivos de registro continuo de eventos, puede ahorrar en los costos de almacenamiento asociados a los archivos de registro. En lugar de almacenar un único archivo de registro, potencialmente grande, se almacenan archivos de registro más pequeños y fáciles de administrar, lo que puede resultar más rentable, especialmente para trabajos complejos o de larga duración.

En un entorno nuevo, los usuarios pueden habilitar de forma explícita la acumulación de registros mediante:

```
'—conf': 'spark.eventLog.rolling.enabled=true'
```

o

```
'—conf': 'spark.eventLog.rolling.enabled=true —conf
spark.eventLog.rolling.maxFileSize=128m'
```

Cuando se activan los registros acumulativos, `spark.eventLog.rolling.maxFileSize` especifica el tamaño máximo del archivo de registro de eventos antes de transferirlo. El valor por defecto de este parámetro opcional, si no se especifica, es 128 MB. El mínimo es de 10 MB.

La suma máxima de todos los archivos de eventos acumulados generados es de 2 GB. Para los AWS Glue trabajos que no admiten registros continuos, el tamaño máximo del archivo de eventos de registro que admite SparkUI es de 0,5 GB.

Puede desactivar los registros acumulativos de un trabajo de streaming al introducir una configuración adicional. Tenga en cuenta que el mantenimiento de archivos de registro muy grandes puede resultar costoso.

Para desactivar los registros continuos, proporcione la siguiente configuración:

```
'--spark-ui-event-logs-path': 'true',
'--conf': 'spark.eventLog.rolling.enabled=false'
```

Lanzamiento del servidor de historial de Spark

Puede utilizar un servidor de historial de Spark para visualizar los registros de Spark en su propia infraestructura. Puede ver en la consola de AWS Glue las mismas visualizaciones del trabajo de AWS Glue que se ejecute en las versiones 4.0 o posteriores deAWS Glue , con registros generados en formato estándar (en lugar de tradicional). Para obtener más información, consulte the section called “Monitorización con la interfaz de usuario de Spark”.

Puede lanzar el servidor de historial de Spark mediante una plantilla AWS CloudFormation que aloja el servidor en una instancia EC2 o lanzar localmente mediante Docker.

Temas

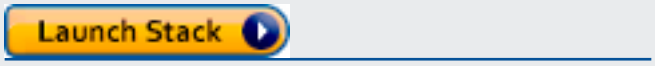
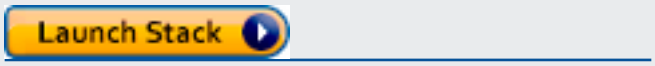
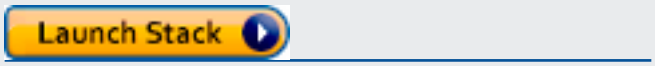
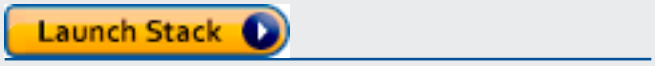
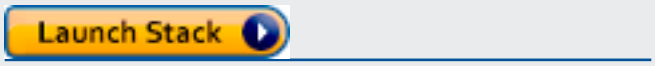
- Lanzamiento del servidor de historial de Spark y visualización de la interfaz de usuario de Spark mediante AWS CloudFormation
- Lanzamiento del servidor de historial de Spark y visualización de la interfaz de usuario de Spark mediante Docker

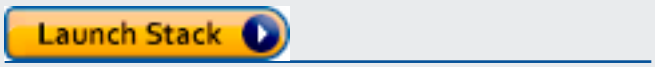
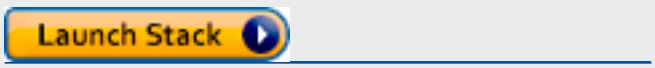
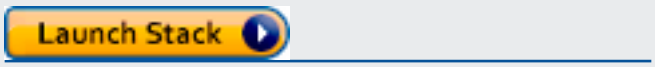
Lanzamiento del servidor de historial de Spark y visualización de la interfaz de usuario de Spark mediante AWS CloudFormation

Puede utilizar una plantilla de AWS CloudFormation para iniciar el servidor de historial de Apache Spark y ver la interfaz de usuario web de Spark. Estas plantillas son ejemplos que debe modificar para satisfacer sus requisitos.

Para iniciar el servidor de historial de Spark y ver la interfaz de usuario de Spark mediante AWS CloudFormation

1. Elija uno de los botones Lanzar pila de la siguiente tabla. Esto lanza la pila en la consola de AWS CloudFormation.

Región	iniciar
US East (Ohio)	
Este de EE. UU. (Norte de Virginia)	
Oeste de EE. UU. (Norte de California)	
Oeste de EE. UU. (Oregón)	
África (Ciudad del Cabo)	
Asia-Pacífico (Hong Kong)	
Asia-Pacífico (Bombay)	
Asia-Pacífico (Osaka)	
Asia-Pacífico (Seúl)	
Asia-Pacífico (Singapur)	
Asia-Pacífico (Sídney)	
Asia-Pacífico (Tokio)	
Canadá (centro)	
Europa (Fráncfort)	

Región	iniciar
Europa (Irlanda)	
Europa (Londres)	
Europa (Milán)	
Europa (París)	
Europa (Estocolmo)	
Medio Oriente (Baréin)	
América del Sur (São Paulo)	

2. En la página Especificar plantilla elija Siguiente.
3. En la página Especificar detalles de pila, escriba el nombre de la pila. Introduzca información adicional en Parámetros.

   a. Configuración de la interfaz de usuario de Spark

      Proporcione la información siguiente:

      - Rango de direcciones IP: el rango de direcciones IP que se puede utilizar para ver la interfaz de usuario de Spark. Si desea restringir el acceso desde un rango de direcciones IP específico, debe utilizar un valor personalizado.
      - Puerto de servidor de historial: el puerto para la interfaz de usuario de Spark. Puede utilizar el valor predeterminado.
      - Directorio de registro de eventos: elija la ubicación en la que se almacenan los registros de eventos de Spark desde el trabajo o los puntos de enlace de desarrollo de AWS Glue. Debe utilizar **`s3a://`** para el esquema de ruta de los registros de eventos.
      - Ubicación del paquete de Spark: puede utilizar el valor predeterminado.

- Ruta del almacén de claves: ruta del almacén de claves SSL/TLS para HTTPS. Si desea utilizar un archivo de almacén de claves personalizado, puede especificar la ruta de S3 `s3://path_to_your_keystore_file` aquí. Si deja este parámetro vacío, se genera y utiliza un almacén de claves basado en certificados autofirmados.
- Contraseña del almacén de claves: ingrese una contraseña del almacén de claves SSL/TLS para HTTPS.

b. Configuración de instancias EC2

Proporcione la información siguiente:

- Tipo de instancia: el tipo de instancia de Amazon EC2 que aloja el servidor de historial de Spark. Dado que esta plantilla lanza una instancia de Amazon EC2 en su cuenta, el costo de Amazon EC2 se cobrará en su cuenta por separado.
- Último ID de AMI: el ID de AMI de Amazon Linux 2 para la instancia del servidor de historial de Spark. Puede utilizar el valor predeterminado.
- ID de VPC: el ID de la nube privada virtual (VPC) para la instancia del servidor de historial de Spark. Puede utilizar cualquiera de las VPC disponibles en su cuenta: no se recomienda utilizar una VPC predeterminada con una ACL de red predeterminada . Para obtener más información, consulte VPC predeterminada y subredes predeterminadas y Creación de una VPC en la Guía del usuario de Amazon VPC.
- ID de subred: el ID de la instancia del servidor de historial de Spark. Puede utilizar cualquiera de las subredes de su VPC. Debe poder acceder a la red desde su cliente a la subred. Si desea obtener acceso a a través de Internet, debe utilizar una subred pública que tenga la gateway de internet en la tabla de ruteo.

c. Elija Siguiente.

4. En la página Configurar las opciones de pila, para utilizar las credenciales de usuario actuales para determinar cómo CloudFormation puede crear, modificar o eliminar recursos de la pila, elija Próximo. También puede especificar un rol en la sección Permisos que utilizará en lugar de los permisos de usuario actuales y, a continuación, elija Siguiente.
5. En la página Revisar, examine la plantilla.

   Elija Confirmo que AWS CloudFormation puede crear recursos de IAM y, a continuación, elija Crear pila.
6. Espere a que se cree la pila.
7. Haga clic en la pestaña Salidas.

a. Copie la URL de SparkUiPublicUrl si está utilizando una subred pública.

b. Copie la URL de SparkUiPrivateUrl si está utilizando una subred privada.

8. Abra un navegador web y pegue la URL. Esto le permite obtener acceso al servidor mediante HTTPS en el puerto especificado. Es posible que el navegador no reconozca el certificado del servidor, en cuyo caso tendrá que anular su protección y continuar.

Lanzamiento del servidor de historial de Spark y visualización de la interfaz de usuario de Spark mediante Docker

Si prefiere el acceso local (no tener una instancia EC2 para el servidor de historial de Apache Spark), también puede utilizar Docker para iniciar el servidor de historial de Apache Spark y ver la interfaz de usuario de Spark localmente. Este Dockerfile es un ejemplo que debe modificar para satisfacer sus requisitos.

Requisitos previos

Para obtener información acerca de cómo instalar Docker en su portátil, consulte la comunidad de Docker Engine.

Iniciar el servidor de historial de Spark y ver la interfaz de usuario de Spark localmente mediante Docker

1. Descargue archivos desde GitHub.

   Descargue Dockerfile y `pom.xml` de las muestras de código de AWS Glue.

2. Determine si desea utilizar las credenciales de usuario o credenciales de usuario federado para acceder a AWS.

   - Para utilizar las credenciales de usuario actuales para acceder a AWS, obtenga los valores que se van a utilizar para `AWS_ACCESS_KEY_ID` y `AWS_SECRET_ACCESS_KEY` en el comando `docker run`. Para obtener más información, consulte Administración de claves de acceso para usuarios de IAM en la Guía del usuario de IAM de .
   - Para utilizar usuarios federados SAML 2.0 para acceder a AWS, obtenga los valores de `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY` y `AWS_SESSION_TOKEN`. Para obtener más información, consulte Solicitud de credenciales de seguridad temporales

3. Determine la ubicación del directorio de registro de eventos que se utilizará en el comando `docker run`.

4. Cree la imagen de Docker con los archivos del directorio local y utilice el nombre `glue/sparkui` y la etiqueta `latest`.

```
$ docker build -t glue/sparkui:latest .
```

5. Cree e inicie el contenedor de docker.

   En los siguientes comandos, utilice los valores obtenidos anteriormente en los pasos 2 y 3.

   a. Para crear el contenedor de docker mediante las credenciales de usuario, utilice un comando similar al siguiente

```
docker run -itd -e SPARK_HISTORY_OPTS="$SPARK_HISTORY_OPTS -
Dspark.history.fs.logDirectory=s3a://path_to_eventlog
 -Dspark.hadoop.fs.s3a.access.key=AWS_ACCESS_KEY_ID -
Dspark.hadoop.fs.s3a.secret.key=AWS_SECRET_ACCESS_KEY"
 -p 18080:18080 glue/sparkui:latest "/opt/spark/bin/spark-class
 org.apache.spark.deploy.history.HistoryServer"
```

   b. Para crear el contenedor de docker mediante credenciales temporales, utilice `org.apache.hadoop.fs.s3a.TemporaryAWSCredentialsProvider` como proveedor, y proporcione los valores de credenciales obtenidos en el paso 2. Para obtener más información, consulte Uso de credenciales de sesión con TemporaryAWSCredentialsProvider en la documentación de Hadoop: integración con Amazon Web Services.

```
docker run -itd -e SPARK_HISTORY_OPTS="$SPARK_HISTORY_OPTS -
Dspark.history.fs.logDirectory=s3a://path_to_eventlog
 -Dspark.hadoop.fs.s3a.access.key=AWS_ACCESS_KEY_ID -
Dspark.hadoop.fs.s3a.secret.key=AWS_SECRET_ACCESS_KEY
 -Dspark.hadoop.fs.s3a.session.token=AWS_SESSION_TOKEN
 -
Dspark.hadoop.fs.s3a.aws.credentials.provider=org.apache.hadoop.fs.s3a.TemporaryAWSCred
 -p 18080:18080 glue/sparkui:latest "/opt/spark/bin/spark-class
 org.apache.spark.deploy.history.HistoryServer"
```

> **Note**
>
> Estos parámetros de configuración proceden del Módulo Hadoop-AWS. Es posible que tenga que agregar configuración específica en función del caso

de uso. Por ejemplo, los usuarios de regiones aisladas tendrán que configurar `spark.hadoop.fs.s3a.endpoint`.

6. Abra `http://localhost:18080` en el navegador para ver la interfaz de usuario de Spark en forma local.

Supervisión con Información de ejecuciones de trabajos de AWS Glue

AWS GlueLa información sobre la ejecución de los trabajos es una función AWS Glue que simplifica la depuración y la optimización de los trabajos. AWS Glue AWS Glueproporciona la interfaz de usuario de Spark y CloudWatch registros y métricas para monitorear tus AWS Glue trabajos. Con esta función, obtienes la siguiente información sobre la ejecución de tu AWS Glue trabajo:

- Número de línea del script del trabajo de AWS Glue con error.
- Última acción de Spark que se ejecutó en el plan de consultas de Spark justo antes del error del trabajo.
- Eventos de excepción de Spark relacionados con el error que se muestran en un flujo de registro ordenado por tiempo.
- Análisis de causa raíz y acción recomendada (por ejemplo, ajustar el script) para solucionar el problema.
- Eventos comunes de Spark (mensajes de registro relacionados con una acción de Spark) con una acción recomendada que aborda la causa raíz.

Puede acceder a toda esta información mediante dos nuevos flujos de registro en los CloudWatch registros de sus AWS Glue trabajos.

Requisitos

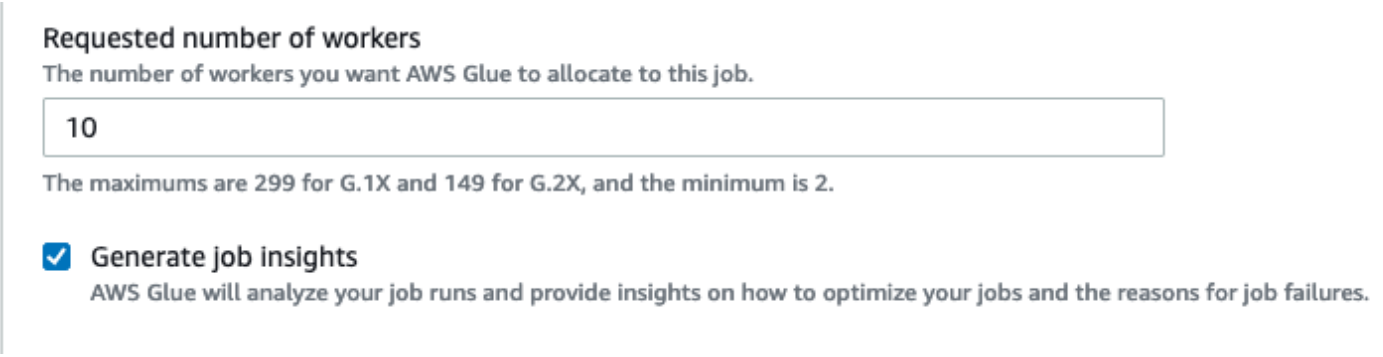
La AWS Glue función de información sobre la ejecución de los trabajos está disponible para AWS Glue las versiones 2.0, 3.0 y 4.0. Puede seguir la guía de migración para actualizar los trabajos existentes desde versiones anteriores de AWS Glue.

Permite obtener información sobre la ejecución de tareas para un trabajo de AWS Glue ETL

Puede habilitar Información de ejecuciones de trabajos a través de AWS Glue Studio o la CLI.

AWS Glue Studio

Al crear un trabajo mediante AWS Glue Studio, puede habilitar o desactivar Información de ejecuciones de trabajos en la pestaña Job Details (Detalles del trabajo). Comprueba que la casilla Generar información sobre el trabajo esté seleccionada.



Línea de comandos

Si crea un trabajo a través de la CLI, puede iniciar una ejecución de trabajos con un único nuevo parámetro de trabajo: `--enable-job-insights = true`.

De forma predeterminada, los flujos de registro de Información de ejecuciones de trabajos se crean en el mismo grupo de registros predeterminado que utiliza el registro continuo de AWS Glue, es decir, `/aws-glue/jobs/logs-v2/`. Puede configurar un nombre de grupo de registros personalizado, filtros de registros y configuraciones de grupos de registros con el mismo conjunto de argumentos que el registro continuo. Para obtener más información, consulte Habilitación del registro continuo para trabajos de AWS Glue.

Al acceder al trabajo, ejecute flujos de registro de información CloudWatch

Con la característica Información de ejecuciones de trabajos habilitada, es posible que haya dos flujos de registro creados cuando falla la ejecución de un trabajo. Cuando un trabajo finaliza correctamente, no se genera ninguno de los flujos.

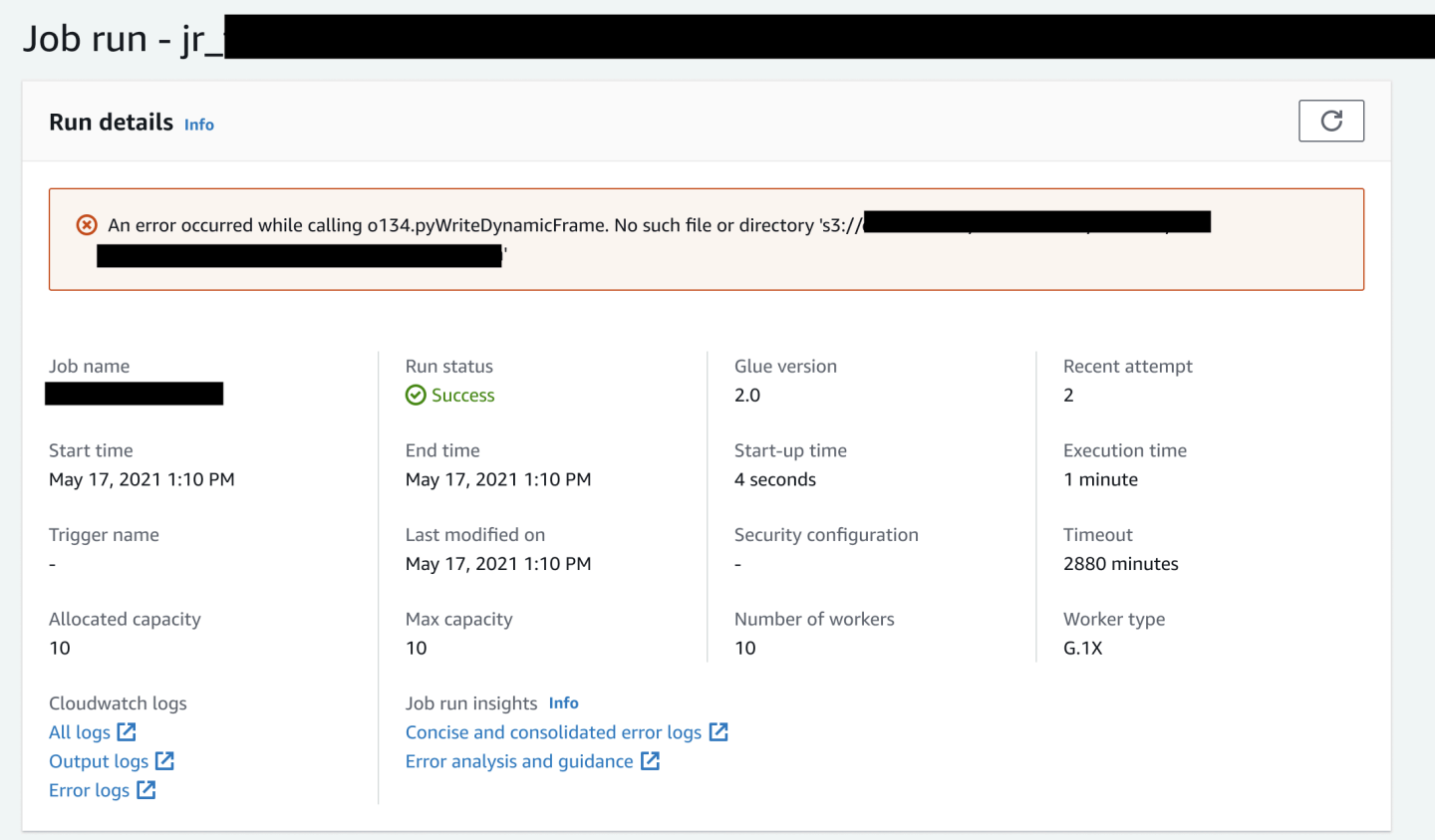
1. Flujo de registro de análisis de excepciones: `<job-run-id>-job-insights-rca-driver`. Este flujo proporciona lo siguiente:
   - Número de línea del script del trabajo de AWS Glue que causó el error.
   - Última acción de Spark que se ejecutó en el plan de consultas de Spark (DAG).
   - Eventos concisos ordenados por tiempo del controlador de Spark y ejecutores relacionados con la excepción. Puede encontrar detalles como mensajes de error completos, la tarea de Spark

con error y el ID de los ejecutores que le ayudan a centrarse en el flujo de registro del ejecutor específico para llevar a cabo una investigación en profundidad si es necesario.

2. Flujo de información basado en reglas:
   - Análisis de causa raíz y recomendaciones sobre cómo corregir los errores (por ejemplo, utilizar un parámetro de trabajo específico para optimizar el rendimiento).
   - Los eventos relevantes de Spark sirven de base para el análisis de causa raíz y una acción recomendada.

> **Note**
>
> El primer flujo solo existirá si hay eventos de excepción de Spark disponibles para una ejecución de trabajo con error, mientras que el segundo flujo solo existirá si hay información disponible para la ejecución del trabajo con error. Por ejemplo, si el trabajo finaliza correctamente, no se generará ninguno de los flujos; si el trabajo falla, pero no hay ninguna regla definida por el servicio que pueda coincidir con el escenario de error, solo se generará el primer flujo.

Si el trabajo se crea en AWS Glue Studio, los enlaces a los flujos anteriores también están disponibles en la pestaña Job run details (Detalles de ejecuciones de trabajos, es decir, Información de ejecuciones de trabajos) como “Concise and consolidated error logs” (Registros de errores concisos y consolidados) y “Error analysis and guidance” (Análisis de errores y orientación).



Ejemplo de Información de ejecuciones de trabajos de AWS Glue

En esta sección, mostramos un ejemplo de cómo la característica Información de ejecuciones de trabajos puede ayudarle a resolver un problema en un trabajo con errores. En este ejemplo, un usuario olvidó importar el módulo obligatorio (tensorflow) en un trabajo de AWS Glue para analizar y crear un modelo de machine learning en sus datos.

```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job
from pyspark.sql.types import *
from pyspark.sql.functions import udf,col

args = getResolvedOptions(sys.argv, ['JOB_NAME'])

sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
```

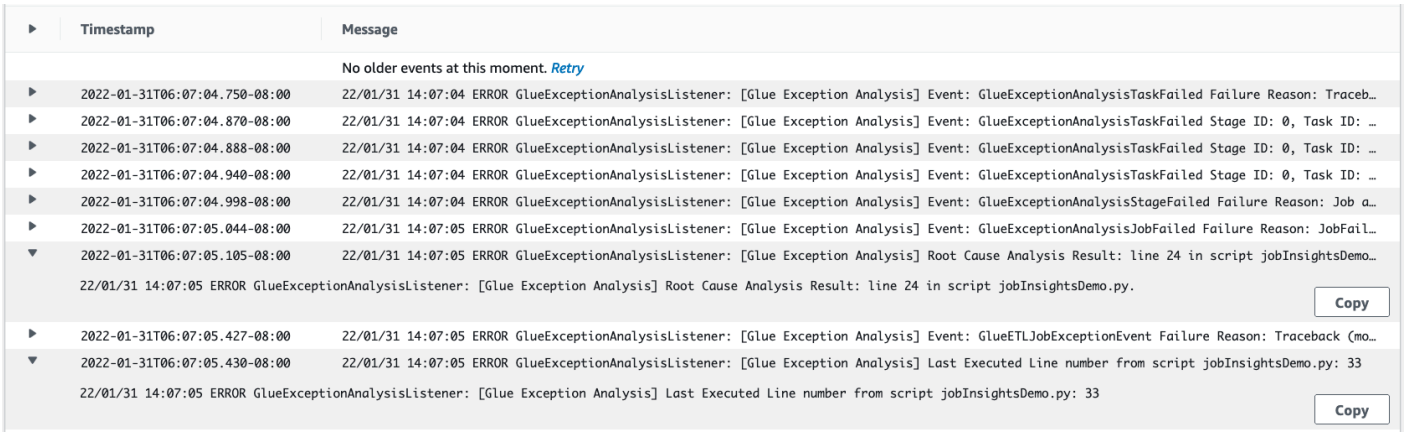
```
job = Job(glueContext)
job.init(args['JOB_NAME'], args)

data_set_1 = [1, 2, 3, 4]
data_set_2 = [5, 6, 7, 8]

scoresDf = spark.createDataFrame(data_set_1, IntegerType())

def data_multiplier_func(factor, data_vector):
    import tensorflow as tf
    with tf.compat.v1.Session() as sess:
        x1 = tf.constant(factor)
        x2 = tf.constant(data_vector)
        result = tf.multiply(x1, x2)
        return sess.run(result).tolist()

data_multiplier_udf = udf(lambda x:data_multiplier_func(x, data_set_2),
 ArrayType(IntegerType(),False))
factoredDf = scoresDf.withColumn("final_value", data_multiplier_udf(col("value")))
print(factoredDf.collect())
```

Sin la característica Información de ejecuciones de trabajos, como el trabajo falla, solo verá este mensaje de Spark:

```
An error occurred while calling o111.collectToPython. Traceback (most
recent call last):
```

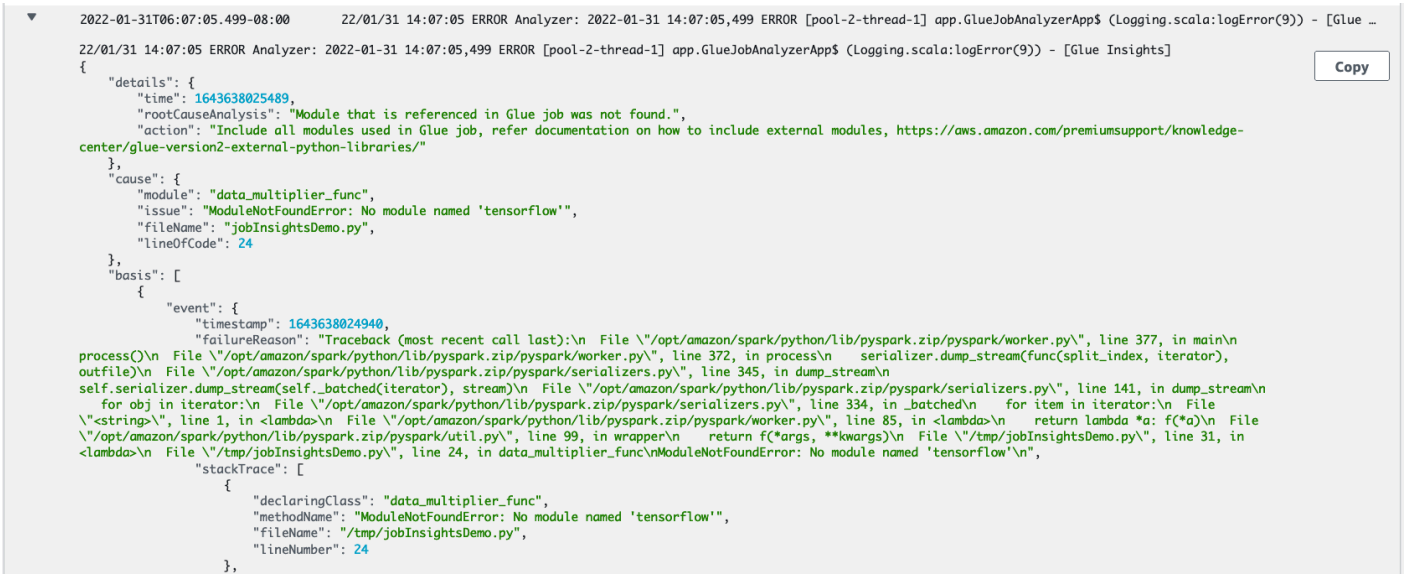
El mensaje es ambiguo y limita la experiencia de depuración. En este caso, esta función le proporciona información adicional sobre dos flujos de CloudWatch registro:

1. El flujo de registro `job-insights-rca-driver`:
   - Eventos de excepción: este flujo de registro proporciona los eventos de excepción de Spark relacionados con el error recopilado por el controlador de Spark y los distintos empleados distribuidos. Estos eventos le ayudan a comprender la propagación ordenada por tiempo de la excepción, ya que el código con error se ejecuta en las tareas, los ejecutores y las etapas de Spark distribuidos en los empleados de AWS Glue.
   - Números de línea: este flujo de registro identifica la línea 21, que hizo la llamada para importar el módulo de Python que falta que causó el error; también identifica la línea 24, la llamada a la acción de Spark `collect()`, como última línea ejecutada del script.



2. El flujo de registro `job-insights-rule-driver`:

   - Causa raíz y recomendación: además del número de línea y el último número de línea ejecutado para el error del script, este flujo de registro muestra el análisis de causa raíz y la recomendación para que siga el documento de AWS Glue y configure los parámetros de trabajo necesarios para utilizar un módulo de Python adicional en su trabajo de AWS Glue.
   - Evento base: este flujo de registro también muestra el evento de excepción de Spark que se evaluó con la regla definida por el servicio para inferir la causa raíz y proporcionar una recomendación.



Monitoreo de con Amazon CloudWatch

Puede monitorear AWS Glue mediante Amazon CloudWatch, que recopila y procesa los datos sin procesar de AWS Glue en métricas legibles y casi en tiempo real. Estas estadísticas se registran durante un periodo de dos semanas, de forma que pueda disponer de información histórica y

obtener una mejor perspectiva sobre el desempeño de la aplicación o servicio web. De forma predeterminada, los datos de las métricas de AWS Glue se envían a CloudWatch en forma automática. Para obtener más información, consulte ¿Qué es Amazon CloudWatch? en la Guía del usuario de Amazon Cloudwatch, y Métricas de AWS Glue.

Registro continuo

AWS Glue también admite el registro continuo en tiempo real para trabajos de AWS Glue. Cuando el registro continuo está habilitado para un trabajo, puede ver los registros en tiempo real en la consola de AWS Glue o en el panel de la consola de CloudWatch. Para obtener más información, consulte Registro continuo para trabajos de AWS Glue.

Métricas de observabilidad

Cuando las Métricas de observabilidad del trabajo están habilitadas, se generan métricas Amazon CloudWatch adicionales cuando se ejecuta el trabajo. Use las métricas de observabilidad AWS Glue para obtener información sobre lo que sucede en el interior de AWS Glue y mejorar la clasificación y el análisis de los problemas.

Temas

- Supervisión de AWS Glue con métricas de Amazon CloudWatch
- Configuración de alarmas de Amazon CloudWatch en perfiles de trabajo de AWS Glue
- Registro continuo para trabajos de AWS Glue
- Monitorización con métricas de observabilidad de AWS Glue

Supervisión de AWS Glue con métricas de Amazon CloudWatch

Puede proporcionar perfiles y supervisar las operaciones de AWS Glue con el generador de trabajos de AWS Glue. Recopila y procesa los datos sin procesar de AWS Glue para convertirlos en métricas legibles y casi en tiempo real en Amazon CloudWatch. Estas estadísticas se retienen y agregan en CloudWatch, de forma que pueda acceder a información histórica para obtener una mejor perspectiva sobre el rendimiento de su aplicación.

> **Note**
>
> Puede incurrir en cargos adicionales cuando habilita las métricas de trabajos y se crean métricas personalizadas de CloudWatch. Para obtener más información, consulte los precios de Amazon CloudWatch .

Información general de métricas de AWS Glue

Cuando usted interactúa con AWS Glue, este último envía métricas a CloudWatch. Puede ver estas métricas en la consola de AWS Glue (el método preferido), el panel de la consola de CloudWatch o AWS Command Line Interface (AWS CLI).

Para consultar las métricas mediante el panel de la consola de AWS Glue

Puede ver gráficos resumidos o detallados de métricas para un trabajo, o bien gráficos detallados para una ejecución de trabajo.

1. Inicie sesión en la AWS Management Console y abra la consola de AWS Glue en https://console.aws.amazon.com/glue/.
2. En el panel de navegación, seleccione Monitoreo de ejecución de trabajos.
3. En Ejecuciones de trabajos, elija Acciones para detener un trabajo que se ejecute actualmente, ver un trabajo o rebobinar el marcador del trabajo.
4. Seleccione un trabajo y, a continuación, elija Ver detalles de la ejecución para ver información adicional sobre la ejecución del trabajo.

Para ver las métricas a través del panel de consola de CloudWatch

Las métricas se agrupan en primer lugar por el espacio de nombres de servicio y, a continuación, por las diversas combinaciones de dimensiones dentro de cada espacio de nombres.

1. Abra la consola de CloudWatch en https://console.aws.amazon.com/cloudwatch/.
2. En el panel de navegación, seleccione Métricas.
3. Seleccione el espacio de nombres Glue.

Para ver métricas mediante la AWS CLI

- En el símbolo del sistema, ejecute el siguiente comando.

```
aws cloudwatch list-metrics --namespace Glue
```

AWS Glue notifica las métricas para CloudWatch cada 30 segundos y los paneles de métricas de CloudWatch se configuran para mostrarlas cada minuto. Las métricas de AWS Glue representan valores delta que se obtienen de los valores notificados con anterioridad. Si procede, los paneles de

métricas acumulan (suman) los valores de 30 segundos para obtener un valor para el último minuto en su totalidad.

Comportamiento de métricas de AWS Glue para trabajos de Spark

Las métricas AWS Glue se habilitan en la inicialización de un `GlueContext` en un script y suelen actualizarse solo al final de una tarea de Apache Spark. Representan los valores de la suma en todas las tareas Spark hasta el momento.

No obstante, las métricas de Spark que AWS Glue transfiere a CloudWatch suelen ser valores absolutos que representan el estado actual en el momento en que se notifican. AWS Glue los notifica a CloudWatch cada 30 segundos, y los paneles de métricas suelen mostrar la media en los puntos de datos recibidos en el último minuto.

Los nombres de las métricas de AWS Glue están precedidos por uno de los siguientes tipos de prefijo:

- `glue.driver.`: las métricas cuyos nombres comienzan con este prefijo representan las métricas de AWS Glue que provienen de todos los ejecutores en el controlador Spark, o bien las métricas de Spark correspondientes al controlador Spark.
- `glue.`Id de ejecutor`.`: el Id de ejecutor es el número de un ejecutor de Spark especificado. Se corresponde con los ejecutores enumerados en los registros.
- `glue.ALL.`: las métricas cuyos nombres empiezan por este prefijo agregan valores de todos los ejecutores de Spark.

Métricas de AWS Glue

AWS Glue crea perfiles y envía las siguientes métricas a CloudWatch cada 30 segundos, y el panel de métricas de AWS Glue los informa una vez por minuto:

Métrica	Descripción
`glue.driver.aggregate.bytesRead`	El número de bytes leídos desde todos los orígenes de datos por todas las tareas de Spark completadas que se ejecutan en todos los ejecutores. Dimensiones válidas: `JobName` (el nombre del trabajo de AWS Glue), `JobRunId` (el ID de JobRun o `ALL`) y `Type` (recuento).

Métrica	Descripción
	Estadísticas válidas: SUMA. Esta métrica es un valor delta desde el último valor notificado, por lo que en el panel de métricas de AWS Glue, se utiliza una estadística SUMA para la agregación. Unidades: bytes Se puede utilizar para monitorear lo siguiente: • Bytes leídos. • Progreso del trabajo. • Orígenes de datos JDBC. • Problemas de marcadores de trabajos. • Desviación entre ejecuciones de trabajos. Esta métrica se puede utilizar de la misma manera que la métrica `glue.ALL.s3.filesystem.read_bytes`, con la diferencia de que esta métrica se actualiza al final de una tarea de Spark y también captura orígenes de datos que no son S3.

Métrica	Descripción
`glue.driver.aggregate.elapsedTime`	El tiempo transcurrido de ETL en milisegundos (no incluye los tiempos de arranque del trabajo). Dimensiones válidas: `JobName` (el nombre del trabajo de AWS Glue), `JobRunId` (el ID de JobRun o `ALL`) y `Type` (recuento). Estadísticas válidas: SUMA. Esta métrica es un valor delta desde el último valor notificado, por lo que en el panel de métricas de AWS Glue, se utiliza una estadística SUMA para la agregación. Unidad: milisegundos Puede utilizarse para determinar cuánto tiempo se tarda en ejecutar una ejecución de trabajo en promedio. Algunas formas de usar los datos: • Establecimiento de alarmas para los rezagados. • Medición de desviación entre ejecuciones de trabajos.

Métrica	Descripción
`glue.driver.aggregate.numCompletedStages`	Número de etapas completadas en este trabajo. Dimensiones válidas: `JobName` (el nombre del trabajo de AWS Glue), `JobRunId` (el ID de JobRun o `ALL`) y `Type` (recuento). Estadísticas válidas: SUMA. Esta métrica es un valor delta desde el último valor notificado, por lo que en el panel de métricas de AWS Glue, se utiliza una estadística SUMA para la agregación. Unidad: recuento Se puede utilizar para monitorear lo siguiente: • Progreso del trabajo. • Línea de tiempo por etapa de ejecución del trabajo, cuando se correlaciona con otras métricas. Algunas formas de usar los datos: • Identificar etapas exigentes en la ejecución de un trabajo. • Establecer alarmas para picos correlacionados (etapas exigentes) en ejecuciones de trabajos.

Métrica	Descripción
`glue.driver.aggregate.numCompletedTasks`	Número de etapas completadas en el trabajo. Dimensiones válidas: `JobName` (el nombre del trabajo de AWS Glue), `JobRunId` (el ID de JobRun o `ALL`) y `Type` (recuento). Estadísticas válidas: SUMA. Esta métrica es un valor delta desde el último valor notificado, por lo que en el panel de métricas de AWS Glue, se utiliza una estadística SUMA para la agregación. Unidad: recuento Se puede utilizar para monitorear lo siguiente: • Progreso del trabajo. • Paralelismo dentro de una etapa.

Métrica	Descripción
`glue.driver.aggregate.numFailedTasks`	El número de tareas de servicio que han presentado un error. Dimensiones válidas: `JobName` (el nombre del trabajo de AWS Glue), `JobRunId` (el ID de JobRun o `ALL`) y `Type` (recuento). Estadísticas válidas: SUMA. Esta métrica es un valor delta desde el último valor notificado, por lo que en el panel de métricas de AWS Glue, se utiliza una estadística SUMA para la agregación. Unidad: recuento Se puede utilizar para monitorear lo siguiente: • Anomalías de datos que provocan un error en las tareas de trabajo. • Anomalías de clúster que provocan un error en las tareas de trabajo. • Anomalías de scripts que causan un error en las tareas de trabajo. Los datos se pueden usar para establecer alarmas ante mayor cantidad de errores que podrían sugerir anomalías en los datos, clústeres o scripts.

Métrica	Descripción
`glue.driver.aggregate.numKilledTasks`	El número de tareas eliminadas. Dimensiones válidas: `JobName` (el nombre del trabajo de AWS Glue), `JobRunId` (el ID de JobRun o `ALL`) y `Type` (recuento). Estadísticas válidas: SUMA. Esta métrica es un valor delta desde el último valor notificado, por lo que en el panel de métricas de AWS Glue, se utiliza una estadística SUMA para la agregación. Unidad: recuento Se puede utilizar para monitorear lo siguiente: • Irregularidades en el sesgo de datos que dan lugar a excepciones (memoria insuficiente) que eliminan tareas. • Irregularidades en los scripts que dan lugar a excepciones (memoria insuficiente) que eliminan tareas. Algunas formas de usar los datos: • Establecer alarmas ante mayor cantidad de errores que indiquen anomalías en los datos. • Establecer alarmas ante mayor cantidad de errores que indiquen anomalías en los clústeres. • Establecer alarmas ante mayor cantidad de errores que indiquen anomalías en los scripts.

Métrica	Descripción
`glue.driver.aggregate.recordsRead`	Número de registros leídos de todos los orígenes de datos por todas las tareas de Spark completadas que se ejecutan en todos los ejecutores. Dimensiones válidas: `JobName` (el nombre del trabajo de AWS Glue), `JobRunId` (el ID de JobRun o `ALL`) y `Type` (recuento). Estadísticas válidas: SUMA. Esta métrica es un valor delta desde el último valor notificado, por lo que en el panel de métricas de AWS Glue, se utiliza una estadística SUMA para la agregación. Unidad: recuento Se puede utilizar para monitorear lo siguiente: • Registros leídos. • Progreso del trabajo. • Orígenes de datos JDBC. • Problemas de marcadores de trabajos. • Sesgo en trabajos que se ejecutan durante varios días. Esta métrica se puede utilizar de manera similar que la métrica `glue.ALL.s3.filesystem.read_bytes` , con la diferencia de que esta métrica se actualiza al final de una tarea de Spark.

Métrica	Descripción
`glue.driver.aggregate.shuffleBytesWritten`	El número de bytes que escriben todos los ejecutores para mezclar los datos en forma aleatoria entre ellos desde el informe anterior (acumulados por el panel de métricas de AWS Glue como el número de bytes escritos para este propósito durante el minuto anterior). Dimensiones válidas: `JobName` (el nombre del trabajo de AWS Glue), `JobRunId` (el ID de JobRun o `ALL`) y `Type` (recuento). Estadísticas válidas: SUMA. Esta métrica es un valor delta desde el último valor notificado, por lo que en el panel de métricas de AWS Glue, se utiliza una estadística SUMA para la agregación. Unidades: bytes Se puede utilizar para monitorear: mezclas aleatorias de datos en trabajos (combinaciones grandes, agrupados por, repartición, fusión). Algunas formas de usar los datos: • Volver a particionar o descomprimir archivos de entrada grandes antes de procesarlos. • Volver a particionar los datos de manera más uniforme para evitar las teclas de acceso rápido. • Prefiltrar los datos antes de las combinaciones o las operaciones GroupBy (Agrupar por).

Métrica	Descripción
`glue.driver.aggregate.shuffleLocalBytesRead`	El número de bytes que leen todos los ejecutores para mezclar los datos en forma aleatoria entre ellos desde el informe anterior (acumulados por el panel de métricas de AWS Glue como el número de bytes leídos para este propósito durante el minuto anterior) . Dimensiones válidas: `JobName` (el nombre del trabajo de AWS Glue), `JobRunId` (el ID de JobRun o `ALL`) y `Type` (recuento). Estadísticas válidas: SUMA. Esta métrica es un valor delta desde el último valor notificado, por lo que en el panel de métricas de AWS Glue, se utiliza una estadística SUMA para la agregación. Unidades: bytes Se puede utilizar para monitorear: mezclas aleatoria s de datos en trabajos (combinaciones grandes, agrupados por, repartición, fusión). Algunas formas de usar los datos: • Volver a particionar o descomprimir archivos de entrada grandes antes de procesarlos. • Volver a particionar los datos de manera más uniforme con teclas de acceso rápido. • Prefiltrar los datos antes de las combinaciones o las operaciones GroupBy (Agrupar por).

Métrica	Descripción
`glue.driver.BlockManager.disk.diskSpaceUsed_MB`	Número de megabytes de espacio en disco utilizado en todos los ejecutores. Dimensiones válidas: `JobName` (el nombre del trabajo de AWS Glue), `JobRunId` (el ID de JobRun o `ALL`), y `Type` (calibre). Estadísticas válidas: promedio. Esta es una métrica de Spark, notificada como un valor absoluto. Unidades: megabytes Se puede utilizar para monitorear lo siguiente: • Espacio en disco utilizado para bloques que representan particiones RDD almacenadas en caché. • Espacio en disco utilizado para bloques que representan salidas de mezclas aleatorias intermedias. • Espacio en disco utilizado para bloques que representan emisiones. Algunas formas de usar los datos: • Identificar errores de trabajo debido a mayor uso del disco. • Identificar grandes particiones que resulten en desbordamiento o mezclado aleatorio. • Aumentar la capacidad de DPU aprovisionada para corregir estos problemas.

Métrica	Descripción
`glue.driver.ExecutorAllocationManager.executors.numberAllExecutors`	El número de ejecutores de trabajo que se ejecutan activamente. Dimensiones válidas: `JobName` (el nombre del trabajo de AWS Glue), `JobRunId` (el ID de JobRun o `ALL`), y `Type` (calibre). Estadísticas válidas: promedio. Esta es una métrica de Spark, notificada como un valor absoluto. Unidad: recuento Se puede utilizar para monitorear lo siguiente: • Actividad del trabajo. • Ejecutores rezagados (solo con algunos ejecutores en proceso de ejecución) • Paralelismo actual a nivel de ejecutor. Algunas formas de usar los datos: • Volver a particionar o descomprimir archivos de entrada grandes por anticipado si el clúster está infrautilizado. • Identificar retrasos en la ejecución de la etapa o del trabajo debido a escenarios rezagados. • • Comparar con NumberMaxNeedExecutors para comprender las tareas pendientes para aprovisio nar más DPU.

Métrica	Descripción
`glue.driver.ExecutorAllocationManager.executors.numberMaxNeededExecutors`	Número máximo de ejecutores de trabajos (en ejecución activa y pendientes) necesarios para satisfacer la carga actual. Dimensiones válidas: `JobName` (el nombre del trabajo de AWS Glue), `JobRunId` (el ID de JobRun o `ALL`), y `Type` (calibre). Estadísticas válidas: máximo. Esta es una métrica de Spark, notificada como un valor absoluto. Unidad: recuento Se puede utilizar para monitorear lo siguiente: • Actividad del trabajo. • Paralelismo actual a nivel de ejecutor y atraso de tareas pendientes aún no programadas debido a que los ejecutores no están disponibles por la capacidad de DPU o ejecutores eliminados/que presentan error. Algunas formas de usar los datos: • Identificar tareas pendientes/atrasadas de la cola de programación. • Identificar retrasos en la ejecución de la etapa o del trabajo debido a escenarios rezagados. • Comparar con NumberAllExecutors para comprender el trabajo atrasado para aprovisionar más DPU. • Aumentar la capacidad de DPU aprovisionada para corregir el retraso del ejecutor pendiente.

Métrica	Descripción

`glue.driver.jvm.heap.usage`
 	
`glue.Id de ejecutor.jvm.heap.usage`
 	
`glue.ALL.jvm.heap.usage`
	La fracción de memoria usada por el montón de JVM para este controlador (escala: 0-1), ejecutor identific ado por el Id de ejecutor o TODOS los ejecutores.
	
	Dimensiones válidas: `JobName` (el nombre del trabajo de AWS Glue), `JobRunId` (el ID de JobRun o `ALL`), y `Type` (calibre).
	
	Estadísticas válidas: promedio. Esta es una métrica de Spark, notificada como un valor absoluto.
	
	Unidad: porcentaje
	
	Se puede utilizar para monitorear lo siguiente:
	
	• Condiciones de memoria insuficiente (OOM) del controlador con `glue.driver.jvm.he ap.usage` .
	• Condiciones de memoria insuficiente (OOM) del ejecutor con `glue.ALL.jvm.heap.usage` .
	
	Algunas formas de usar los datos:
	
	• Identificar las etapas y los ID de los ejecutores que consumen memoria.
	• Identificar los ID y las etapas de los ejecutores rezagados.
	• Identificar una condición de memoria insuficiente (OOM) del controlador.
	• Identificar una condición de falta de memoria (OOM) del ejecutor y obtener el ID de ejecutor correspondiente para poder obtener un seguimien to de pila del registro del ejecutor.

Métrica	Descripción
	• Identificar los archivos o particiones que pueden tener un sesgo de datos, lo que resulta en rezagos o condiciones de memoria insuficiente (OOM).

Métrica	Descripción

`glue.driver.jvm.heap.used`
 	
`glue.`Id de ejecutor`.jvm.heap.used`
 	
`glue.ALL.jvm.heap.used`
	El número de bytes de memoria utilizados por el montón de JVM para el controlador, el ejecutor identificado por Id de ejecutor, o TODOS los ejecutores.
	
	Dimensiones válidas: `JobName` (el nombre del trabajo de AWS Glue), `JobRunId` (el ID de JobRun o `ALL`), y `Type` (calibre).
	
	Estadísticas válidas: promedio. Esta es una métrica de Spark, notificada como un valor absoluto.
	
	Unidades: bytes
	
	Se puede utilizar para monitorear lo siguiente:
	
	• Condiciones de memoria insuficiente (OOM) del controlador.
	• Condiciones de memoria insuficiente (OOM) del ejecutor.
	
	Algunas formas de usar los datos:
	
	• Identificar las etapas y los ID de los ejecutores que consumen memoria.
	• Identificar los ID y las etapas de los ejecutores rezagados.
	• Identificar una condición de memoria insuficiente (OOM) del controlador.
	• Identificar una condición de falta de memoria (OOM) del ejecutor y obtener el ID de ejecutor correspondiente para poder obtener un seguimien to de pila del registro del ejecutor.

Métrica	Descripción
	• Identificar los archivos o particiones que pueden tener un sesgo de datos, lo que resulta en rezagos o condiciones de memoria insuficiente (OOM).

Métrica	Descripción

`glue.driver.s3.filesystem.read_bytes`
 	
`glue.`Id de ejecutor`.s3.filesystem.read_bytes`
 	
`glue.ALL.s3.filesystem.read_bytes`
	El número de bytes leídos desde Amazon S3 por el controlador, un ejecutor identificado por Id de ejecutor, o TODOS los ejecutores desde el informe anterior (acumulados por el panel de métricas de AWS Glue como número de bytes leídos durante el minuto anterior).
	
	Dimensiones válidas: `JobName`, `JobRunId`, y `Type` (calibre).
	
	Estadísticas válidas: SUMA. Esta métrica es un valor delta desde el último valor notificado, por lo que en el panel de métricas de AWS Glue, se utiliza una estadística SUM para la agregación. El área bajo la curva en el panel de métricas de AWS Glue se puede utilizar para comparar visualmente los bytes leídos por dos ejecuciones de trabajos diferentes.
	
	Unidad: bytes.
	
	Se puede utilizar para monitorear lo siguiente:
	
	• Movimiento de datos de ETL.
	• Progreso del trabajo.
	• Problemas de marcadores de trabajos (datos procesados, reprocesados y omitidos).
	• Comparación de lecturas con la tasa de ingesta de orígenes de datos externos.
	• Desviación entre ejecuciones de trabajos.
	
	Los datos resultantes se pueden utilizar para lo siguiente:
	
	• Planificación de la capacidad de DPU.

Métrica	Descripción
	• Establecer alarmas para picos grandes o caídas en los datos leídos para ejecuciones de trabajos y etapas de trabajo.

Métrica	Descripción

`glue.driver.s3.filesystem.write_bytes`
 	
`glue.`Id de ejecutor`.s3.filesystem.write_bytes`
 	
`glue.ALL.s3.filesystem.write_bytes`
	El número de bytes que escribe el controlador en Amazon S3, un ejecutor identificado por Id de ejecutor, o TODOS los ejecutores desde el informe anterior (agregados por el panel de métricas de AWS Glue como número de bytes escritos durante el minuto anterior).
	
	Dimensiones válidas: `JobName`, `JobRunId`, y `Type` (calibre).
	
	Estadísticas válidas: SUMA. Esta métrica es un valor delta desde el último valor notificado, por lo que en el panel de métricas de AWS Glue, se utiliza una estadística SUM para la agregación. El área bajo la curva en el panel de métricas de AWS Glue se puede utilizar para comparar visualmente los bytes escritos por dos ejecuciones de trabajos diferentes.
	
	Unidades: bytes
	
	Se puede utilizar para monitorear lo siguiente:
	
	• Movimiento de datos de ETL.
	• Progreso del trabajo.
	• Problemas de marcadores de trabajos (datos procesados, reprocesados y omitidos).
	• Comparación de lecturas con la tasa de ingesta de orígenes de datos externos.
	• Desviación entre ejecuciones de trabajos.
	
	Algunas formas de usar los datos:
	
	• Planificación de la capacidad de DPU.

Métrica	Descripción
	• Establecer alarmas para picos grandes o caídas en los datos leídos para ejecuciones de trabajos y etapas de trabajo.
`glue.driver.streaming.numRecords`	El número de registros que se reciben en un microlote. Esta métrica solo está disponible para trabajos de streaming de AWS Glue con la versión AWS Glue 2.0 y superior. Dimensiones válidas: `JobName` (el nombre del trabajo de AWS Glue), `JobRunId` (el ID de JobRun o `ALL`), y `Type` (recuento). Estadísticas válidas: suma, máximo, mínimo, promedio, percentilo Unidad: recuento Se puede utilizar para monitorear lo siguiente: • Registros leídos. • Progreso del trabajo.

Métrica	Descripción
`glue.driver.streaming.batchProcessingTimeInMs`	El tiempo que se tarda en procesar los lotes en milisegundos. Esta métrica solo está disponible para trabajos de streaming de AWS Glue con la versión AWS Glue 2.0 y superior. Dimensiones válidas: `JobName` (el nombre del trabajo de AWS Glue), `JobRunId` (el ID de JobRun o `ALL`), y `Type` (recuento). Estadísticas válidas: suma, máximo, mínimo, promedio, percentilo Unidad: recuento Se puede utilizar para monitorear lo siguiente: • Progreso del trabajo. • Rendimiento del script.

Métrica	Descripción

`glue.driver.system.cpuSystemLoad`
 	
`glue.`Id de ejecutor`.system.cpuSystemLoad`
 	
`glue.ALL.system.cpuSystemLoad`
	Fracción de la carga del sistema de CPU usada (escala: 0-1) por el controlador, un ejecutor identificado por Id de ejecutor, o TODOS los ejecutores.
	
	Dimensiones válidas: `JobName` (el nombre del trabajo de AWS Glue), `JobRunId` (el ID de JobRun o `ALL`), y `Type` (calibre).
	
	Estadísticas válidas: promedio. Esta métrica se notifica como un valor absoluto.
	
	Unidad: porcentaje
	
	Se puede utilizar para monitorear lo siguiente:
	
	• Carga de CPU del controlador.
	• Carga de CPU del ejecutor.
	• Detección de ejecutores o etapas vinculados a la CPU o vinculados a entradas/salidas en un trabajo.
	
	Algunas formas de usar los datos:
	
	• Planificación de la capacidad de DPU junto con métricas de entrada/salida (bytes de lectura/bytes de mezcla aleatoria, paralelismo de tareas) y la métrica del número máximo de ejecutores necesarios.
	• Identificación de la relación vinculada a CPU y entrada/salida. Esto permite la repartición y el aumento de la capacidad aprovisionada para trabajos de ejecución prolongada con conjuntos de datos que pueden dividirse y ocupan menor capacidad de uso de la CPU.

Dimensiones de las métricas de AWS Glue

Las métricas de AWS Glue utilizan el espacio de nombres de AWS Glue y proporcionan métricas para las siguientes dimensiones:

Dimensión	Descripción
`JobName`	Esta dimensión filtra las métricas de todas las ejecuciones de trabajos de un trabajo de AWS Glue específico.
`JobRunId`	Esta dimensión filtra las métricas de ejecución de un trabajo de AWS Glue específico por ID de JobRun, o `ALL`.
`Type`	Esta dimensión filtra las métricas por `count` (un número acumulado) o `gauge` (un valor en un punto en el tiempo).

Para más información, consulte la Guía del usuario de Amazon CloudWatch.

Configuración de alarmas de Amazon CloudWatch en perfiles de trabajo de AWS Glue

Las métricas de AWS Glue también están disponibles en Amazon CloudWatch. Puede configurar alarmas en cualquier métrica de AWS Glue para trabajos programados.

A continuación se muestran algunas situaciones comunes de configuración de alarmas:

- Trabajos con problemas de falta de memoria (OOM): establezca una alarma cuando el uso de la memoria supere la media normal del controlador o un ejecutor de un trabajo de AWS Glue.
- Ejecutores rezagados: establezca una alarma cuando el número de ejecutores caiga por debajo de un determinado umbral durante un largo periodo de tiempo en un trabajo de AWS Glue.
- Reprocesamiento o tareas pendientes de datos: compare las métricas de trabajos individuales en un flujo de trabajo mediante una expresión matemática de CloudWatch. A continuación, puede activar una alarma en el valor de expresión resultante (como la relación de bytes escritos por un trabajo y bytes leídos por un trabajo posterior).

Para obtener instrucciones detalladas acerca de cómo establecer alarmas, consulte Crear o editar una alarma de CloudWatch Alarm en la Guía del usuario de Amazon CloudWatch Events.

Para monitorear y depurar escenarios con CloudWatch, consulte Monitorización y depuración de trabajo.

Registro continuo para trabajos de AWS Glue

AWS Glue ofrece un registro continuo en tiempo real de los trabajos de AWS Glue. Puede ver registros de trabajos de Apache Spark en tiempo real en Amazon CloudWatch, incluidos registros del controlador, registros de tareas y una barra de progreso de un trabajo de Apache Spark. La posibilidad de ver registros en tiempo real le proporciona una mejor perspectiva de la ejecución de trabajo.

Cuando inicia un trabajo de AWS Glue, envía la información de registro en tiempo real a CloudWatch (cada cinco segundos y antes de que finalice cada ejecutor) después de que la aplicación Spark empiece a ejecutarse. Puede ver los registros en la consola de AWS Glue o el panel de la consola de CloudWatch.

La función de registro continuo incluye las siguientes capacidades:

- Registro continuo
- Un registro de script personalizado para registrar los mensajes específicos de la aplicación.
- Una barra de progreso de la consola para realizar un seguimiento del estado de ejecución del trabajo de AWS Glue actual.

Para obtener información sobre cómo se soporta el registro continuo en AWS Glue versión 2.0, consulte Ejecución de trabajos de ETL de Spark con tiempos de inicio reducidos.

Puede restringir el acceso a grupos de CloudWatch Logs o flujos para que los roles de IAM lean los registros. Para obtener más información sobre cómo restringir el acceso, consulte Utilizar políticas basadas en identidad (políticas de IAM) para CloudWatch Logs en la documentación de CloudWatch.

> **Note**
>
> Puede incurrir en cargos adicionales cuando habilita el registro continuo y se crean eventos de registro de CloudWatch adicionales. Para obtener más información, consulte los precios de Amazon CloudWatch .

Temas

- Habilitación del registro continuo para trabajos de AWS Glue
- Visualización del registro continuo para trabajos de AWS Glue

Habilitación del registro continuo para trabajos de AWS Glue

Puede habilitar el registro continuo a través de la consola de AWS Glue o a través de la AWS Command Line Interface (AWS CLI).

Puede habilitar el registro continuo al crear un trabajo nuevo, editar un trabajo existente o habilitarlo a través de AWS CLI.

También puede especificar opciones de configuración personalizadas, como el nombre del grupo de registro de Amazon CloudWatch, el prefijo de flujo de registro de CloudWatch antes del ID de controlador/ID de ejecutor de ejecución del trabajo de AWS Glue, así como el patrón de conversión de registro para los mensajes de registro. Estas configuraciones le ayudan a establecer registros agregados en grupos de registros personalizados de CloudWatch con diferentes políticas de caducidad y a analizarlos con prefijos de flujo de registro personalizados y patrones de conversión.

Temas

- Uso de la AWS Management Console
- Registro de mensajes específicos de aplicación con el registrador de script personalizado
- Habilitación de la barra de progreso para mostrar el progreso del trabajo
- Configuración de seguridad con registro continuo.

Uso de la AWS Management Console

Siga estos pasos para usar la consola para habilitar el registro continuo al crear o editar un trabajo de AWS Glue.

Para crear un nuevo trabajo de AWS Glue con registro continuo

1. Inicie sesión en la AWS Management Console y abra la consola de AWS Glue en https://console.aws.amazon.com/glue/.
2. En el panel de navegación, seleccione Trabajos de ETL.
3. Seleccione Visual ETL.

4. En la pestaña Detalles del trabajo, expanda la sección Propiedades avanzadas.
5. En la sección Registro continuo, seleccione Habilitar registros en CloudWatch.

Para habilitar el registro continuo para un trabajo de AWS Glue que ya tenga

1. Abra la consola de AWS Glue en https://console.aws.amazon.com/glue/.
2. En el panel de navegación, seleccione Trabajos.
3. Elija el trabajo que ya tiene en la lista Jobs (Trabajos).
4. Elija Action (Acción), Edit job (Editar trabajo).
5. En la pestaña Detalles del trabajo, expanda la sección Propiedades avanzadas.
6. En la sección Registro continuo, seleccione Habilitar registros en CloudWatch.

Mediante AWS CLI

Para habilitar el registro continuo debe transferir los parámetros de trabajo a un trabajo de AWS Glue. Transfiera los parámetros de trabajos especiales similares que figuran a continuación a otros parámetros de trabajo de AWS Glue. Para obtener más información, consulte Parámetros de los trabajos de AWS Glue.

```
'--enable-continuous-cloudwatch-log': 'true'
```

Puede especificar un nombre de grupo de registro personalizado de Amazon CloudWatch. Si no se especifica, el nombre predeterminado del grupo de registro es `/aws-glue/jobs/logs-v2/`.

```
'--continuous-log-logGroup': 'custom_log_group_name'
```

Puede especificar un prefijo de flujo de registro personalizado de Amazon CloudWatch. Si no se especifica, el prefijo de flujo de registro predeterminado es el ID de ejecución del trabajo.

```
'--continuous-log-logStreamPrefix': 'custom_log_stream_prefix'
```

Puede especificar un patrón de conversión de registro continuo personalizado. Si no se especifica, el patrón de conversión predeterminado es `%d{yy/MM/dd HH:mm:ss} %p %c{1}: %m%n`. Tenga en cuenta que el patrón de conversión solo se aplica a los registros de controlador y ejecutor. No afecta a la barra de progreso de AWS Glue.

```
'--continuous-log-conversionPattern': 'custom_log_conversion_pattern'
```

Registro de mensajes específicos de aplicación con el registrador de script personalizado

Puede utilizar el registrador de AWS Glue para registrar todos los mensajes específicos de la aplicación en el script que se envían en tiempo real al flujo de registro del controlador.

En el siguiente ejemplo se muestra un script de Python.

```
from awsglue.context import GlueContext
from pyspark.context import SparkContext

sc = SparkContext()
glueContext = GlueContext(sc)
logger = glueContext.get_logger()
logger.info("info message")
logger.warn("warn message")
logger.error("error message")
```

En el siguiente ejemplo se muestra un script de Scala.

```
import com.amazonaws.services.glue.log.GlueLogger

object GlueApp {
  def main(sysArgs: Array[String]) {
    val logger = new GlueLogger
    logger.info("info message")
    logger.warn("warn message")
    logger.error("error message")
  }
}
```

Habilitación de la barra de progreso para mostrar el progreso del trabajo

AWS Glue proporciona una barra de progreso en tiempo real bajo el flujo de registro `JOB_RUN_ID-progress-bar` para comprobar el estado de ejecución de un trabajo de AWS Glue. En la actualidad se admiten únicamente los trabajos que inicializan `glueContext`. Si ejecuta un trabajo de Spark puro sin inicializar `glueContext`, la barra de progreso de AWS Glue no aparece.

La barra de progreso muestra la siguiente actualización del progreso cada 5 segundos.

```
Stage Number (Stage Name): > (numCompletedTasks + numActiveTasks) /
 totalNumOfTasksInThisStage]
```

Configuración de seguridad con registro continuo.

Si se habilita una configuración de seguridad para los registros de CloudWatch, AWS Glue creará un grupo de registro llamado de la siguiente manera para los registros continuos:

```
<Log-Group-Name>-<Security-Configuration-Name>
```

Los grupos de registro predeterminados y personalizados serán de la siguiente manera:

- El grupo de registro continuo predeterminado será `/aws-glue/jobs/logs-v2-`*`<Security-Configuration-Name>`*
- El grupo de registro continuo personalizado será *`<custom-log-group-name>`*`-`*`<Security-Configuration-Name>`*

Debe agregar `logs:AssociateKmsKey` a sus permisos de rol de IAM, si habilita una configuración de seguridad con CloudWatch Logs. Si no se incluye ese permiso, se deshabilitará el registro continuo. Además, para configurar el cifrado de CloudWatch Logs, siga las instrucciones en Cifrar datos de registro en CloudWatch Logs mediante AWS Key Management Service en la Guía del usuario de Amazon CloudWatch Logs.

Para obtener más información sobre cómo crear una configuración de seguridad, consulte Trabajar con configuraciones de seguridad en la consola de AWS Glue.

Visualización del registro continuo para trabajos de AWS Glue

Puede ver registros en tiempo real mediante la consola de AWS Glue o la consola de Amazon CloudWatch.

Para ver registros en tiempo real mediante el panel de la consola de AWS Glue

1. Inicie sesión en la AWS Management Console y abra la consola de AWS Glue en https://console.aws.amazon.com/glue/.
2. En el panel de navegación, seleccione Trabajos.
3. Añada o inicie el trabajo que ya tiene. Elija Action (Acción), Run job (Ejecutar trabajo).

Cuando inicia la ejecución de un trabajo, puede ir a una página que contiene información sobre el trabajo en ejecución:

- La pestaña Logs (Registros) muestra los registros de la aplicación agregados más antiguos.
- La pestaña Continuous logging (Registro continuo) muestra una barra de progreso en tiempo real cuando el trabajo se ejecuta con `glueContext` inicializado.
- La pestaña Continuous logging (Registro continuo) también contiene los registros del controlador, que capturan registros de controlador de Apache Spark en tiempo real y registros de aplicación del script registrados con el registrador de la aplicación de AWS Glue cuando el trabajo se está ejecutando.

4. Para trabajos más antiguos también puede ver los registros en tiempo real en la vista Job History (Historial de trabajos) eligiendo Logs (Registros). Esta acción lo lleva a la consola de CloudWatch que muestra todos los flujos de registro del controlador de Spark, ejecutor y barra de progreso para ese trabajo ejecutado.

Para ver registros en tiempo real mediante el panel de la consola de CloudWatch

1. Abra la consola de CloudWatch en https://console.aws.amazon.com/cloudwatch/.
2. En el panel de navegación, elija Log (Registro).
3. Elija el grupo de registros /aws-glue/jobs/logs-v2/.
4. En el cuadro Filter (Filtro), pegue ID del trabajo en ejecución.

   Puede ver los registros del controlador, los registros del ejecutor y la barra de progreso (si utiliza el filtro estándar).

Monitorización con métricas de observabilidad de AWS Glue

> **Note**
>
> Las métricas de observabilidad de AWS Glue están disponibles en AWS Glue, versión 4.0 y versiones posteriores.

Utilice las métricas de observabilidad de AWS Glue para obtener información sobre lo que ocurre dentro de sus trabajos de AWS Glue for Apache Spark y así mejorar la clasificación y el análisis de

los problemas. Las métricas de observabilidad se visualizan a través de paneles de control Amazon CloudWatch y se pueden utilizar para analizar la causa raíz de los errores y diagnosticar los cuellos de botella en el rendimiento. Puede reducir el tiempo dedicado a depurar los problemas a gran escala para poder centrarse en resolverlos de forma más rápida y eficaz.

La observabilidad de AWS Glue proporciona métricas Amazon CloudWatch clasificadas en los cuatro grupos siguientes:

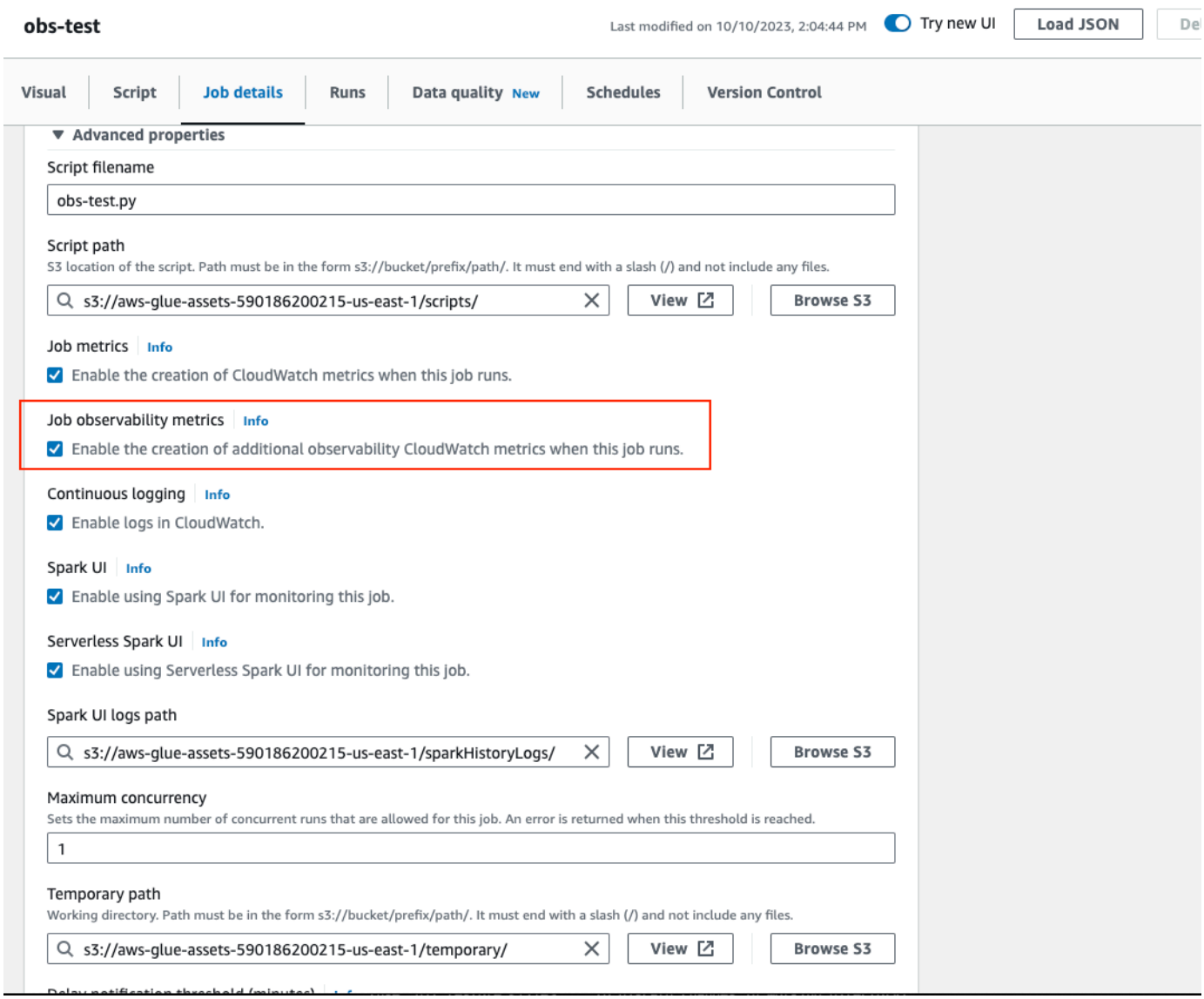
- Fiabilidad (es decir, clases de errores): identifique fácilmente los motivos de error más comunes en un intervalo de tiempo determinado que desee abordar.
- Rendimiento (es decir, asimetría): identifique un obstáculo en el rendimiento y aplique técnicas de ajuste. Por ejemplo, si experimenta una disminución del rendimiento debido a la asimetría de los trabajos, es posible que desee activar Spark Adaptive Query Execution y ajustar el umbral de unión descompensado.
- Rendimiento (es decir, rendimiento por fuente/receptor): supervisa las tendencias de lectura y escritura de datos. También puede configurar alarmas Amazon CloudWatch para detectar anomalías.
- Utilización de los recursos (es decir, el personal, la utilización de la memoria y el disco): encuentre de manera eficiente los trabajos con un bajo uso de la capacidad. Es posible que desee habilitar el escalado automático de AWS Glue para esos trabajos.

Cómo empezar con las métricas de observabilidad de AWS Glue

> **Note**
>
> Las nuevas métricas están habilitadas de forma predeterminada en la consola de AWS Glue Studio.

Para configurar las métricas de observabilidad en AWS Glue Studio:

1. Inicie sesión en la consola AWS Glue y seleccione Trabajos ETL en el menú de la consola.
2. Elija un trabajo haciendo clic en el nombre del trabajo en la sección Sus trabajos.
3. Elija la pestaña Detalles del trabajo.
4. Desplácese hasta la parte inferior y seleccione Propiedades avanzadas y, a continuación, Métricas de observabilidad del trabajo.



Para habilitar las métricas de observabilidad de AWS Glue usando AWS CLI:

- Agregue al mapa `--default-arguments` el siguiente valor-clave en el archivo JSON de entrada:

```
--enable-observability-metrics, true
```

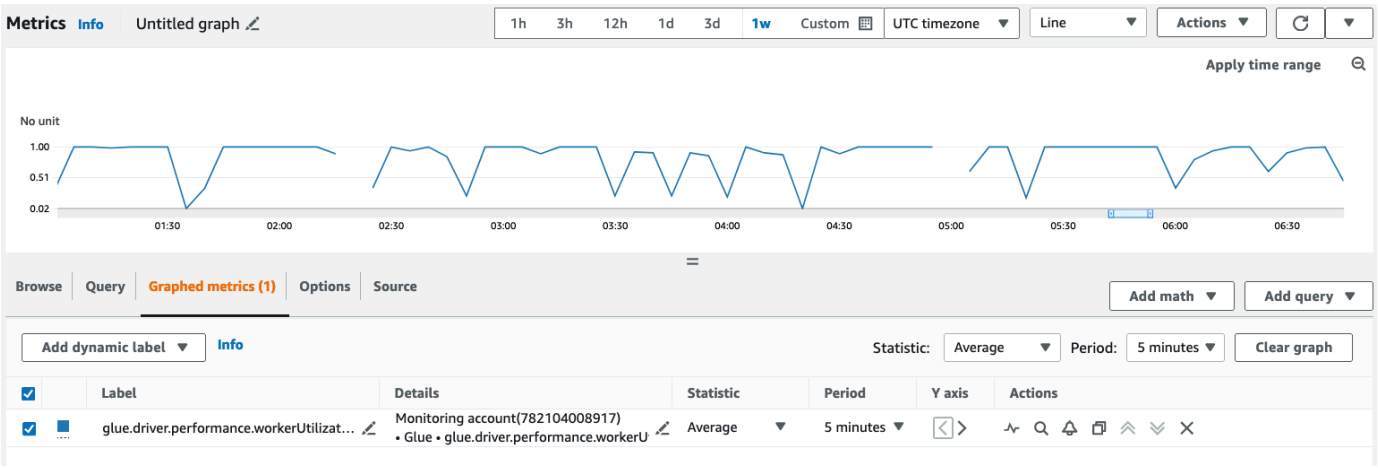
Uso de la observabilidad de AWS Glue

Dado que las métricas de observabilidad de AWS Glue se proporcionan mediante Amazon CloudWatch, puede usar la consola Amazon CloudWatch, AWS CLI, el SDK o la API para consultar los puntos de datos de las métricas de observabilidad. Consulte Uso de observabilidad de Glue para monitorear el uso de los recursos a fin de reducir el costo para ver un ejemplo de caso práctico en el que se utilizan métricas de observabilidad de AWS Glue.

Uso de la observabilidad de AWS Glue en la consola Amazon CloudWatch

Para consultar y visualizar las métricas en la consola Amazon CloudWatch:

1. Abra la consola Amazon CloudWatch y seleccione Todas las métricas.
2. En Espacios de nombres personalizados, seleccione AWS Glue.
3. Elija Métricas de observabilidad de trabajo, Métricas de observabilidad por origen o Métricas de observabilidad por receptor.
4. Busque el nombre de la métrica, el nombre del trabajo o el identificador de ejecución específicos y selecciónelos.
5. En la pestaña Métricas graficadas, configure la estadística, el período y otras opciones que prefiera.



Para consultar una métrica de observabilidad mediante AWS CLI:

1. Cree un archivo JSON de definición de métricas y reemplace `your-Glue-job-name` y `your-Glue-job-run-id`.

```
$ cat multiplequeries.json
```

```
[
    {
        "Id": "avgWorkerUtil_0",
        "MetricStat": {
            "Metric": {
                "Namespace": "Glue",
                "MetricName": "glue.driver.workerUtilization",
                "Dimensions": [
                    {
                        "Name": "JobName",
                        "Value": "<your-Glue-job-name-A>"
                    },
                    {
                        "Name": "JobRunId",
                        "Value": "<your-Glue-job-run-id-A>"
                    },
                    {
                        "Name": "Type",
                        "Value": "gauge"
                    },
                    {
                        "Name": "ObservabilityGroup",
                        "Value": "resource_utilization"
                    }
                ]
            },
            "Period": 1800,
            "Stat": "Minimum",
            "Unit": "None"
        }
    },
    {
        "Id": "avgWorkerUtil_1",
        "MetricStat": {
            "Metric": {
                "Namespace": "Glue",
                "MetricName": "glue.driver.workerUtilization",
                "Dimensions": [
                    {
                        "Name": "JobName",
                        "Value": "<your-Glue-job-name-B>"
                    },
                    {
                        "Name": "JobRunId",
```

```
                        "Value": "<your-Glue-job-run-id-B>"
                    },
                    {
                        "Name": "Type",
                        "Value": "gauge"
                    },
                    {
                        "Name": "ObservabilityGroup",
                        "Value": "resource_utilization"
                    }
                ]
            },
            "Period": 1800,
            "Stat": "Minimum",
            "Unit": "None"
        }
    }
]
```

2. Ejecute el comando `get-metric-data`:

```
$ aws cloudwatch get-metric-data --metric-data-queries file: //multiplequeries.json
 \
     --start-time '2023-10-28T18: 20' \
     --end-time '2023-10-28T19: 10'  \
     --region us-east-1
{
    "MetricDataResults": [
        {
            "Id": "avgWorkerUtil_0",
            "Label": "<your-label-for-A>",
            "Timestamps": [
                "2023-10-28T18:20:00+00:00"
            ],
            "Values": [
                0.06718750000000001
            ],
            "StatusCode": "Complete"
        },
        {
            "Id": "avgWorkerUtil_1",
            "Label": "<your-label-for-B>",
```

```
            "Timestamps": [
                "2023-10-28T18:50:00+00:00"
            ],
            "Values": [
                0.5959183673469387
            ],
            "StatusCode": "Complete"
        }
    ],
    "Messages": []
}
```

Métricas de observabilidad

La observabilidad de AWS Glue crea perfiles y envía las siguientes métricas a Amazon CloudWatch cada 30 segundos, y algunas de estas métricas pueden estar visibles en la página de monitoreo de JobRuns de AWS Glue Studio.

Métrica	Descripción	Categoría
glue.driver.skewness.stage	Categoría métrica: job_performance Spark clasifica la asimetría de la ejecución: esta métrica captura la asimetría de la ejecución, que puede deberse a una asimetría de los datos de entrada o a una transformación (por ejemplo, una unión sesgada). Los valores de esta métrica se sitúan en el rango de [0, infinito [, donde 0 indica la relación entre el tiempo máximo y el tiempo medio de ejecución de las tareas; entre todas las tareas de la etapa, es inferior a un factor de	job_performance

Métrica	Descripción	Categoría
	asimetría de fase determina do. El factor de asimetría de fase predeterminado es 5 y se sobrescribe mediante spark conf: spark.metrics.conf .driver.source.glue.jobperf ormance.skewnessFactor Un valor de asimetría de fase igual a 1 significa que la relación es el doble del factor de asimetría de fase. El valor de la asimetría de fase se actualiza cada 30 segundos para reflejar la asimetría actual. El valor al final de la etapa refleja la asimetría final de la etapa. Dimensiones válidas: JobName (el nombre del trabajo de AWS Glue), JobRunId (el ID de JobRun o ALL), Type (calibre) y ObservAbilityGroup (job_perf ormance) Estadísticas válidas: promedio, máximo, mínimo, percentilo Unidad: recuento	

Métrica	Descripción	Categoría
glue.driver.skewness.job	Categoría métrica: job_performance La asimetría del trabajo es el promedio ponderado de la asimetría de las etapas del trabajo. El promedio ponderado da más peso a las etapas que tardan más en ejecutarse. El propósito de esto evitar el caso extremo en el que una etapa muy asimétrica dure muy poco tiempo en comparación con otras etapas (y, por lo tanto, su asimetría no sea significativa para el rendimiento general del trabajo y no valga la pena hacer el esfuerzo para tratar de corregir esa asimetría). Esta métrica se actualiza al finalizar cada etapa y, por lo tanto, el último valor refleja la asimetría general real del trabajo. Dimensiones válidas: JobName (el nombre del trabajo de AWS Glue), JobRunId (el ID de JobRun o ALL), Type (calibre) y ObservAbilityGroup (job_performance)	job_performance

Métrica	Descripción	Categoría
	Estadísticas válidas: promedio, máximo, mínimo, percentilo Unidad: recuento	
glue.succeed.ALL	Categoría métrica: error Número total de tareas ejecutadas correctamente, para completar el panorama de las categorías de errores Dimensiones válidas: JobName (el nombre del trabajo de AWS Glue), JobRunId (el ID de JobRun o ALL), Type (recuento) y ObservAbilityGroup (error) Estadísticas válidas: SUMA Unidad: recuento	error

Métrica	Descripción	Categoría
glue.error.ALL	Categoría métrica: error Número total de errores al momento de ejecutar el trabajo, para completar el panorama de las categorías de errores Dimensiones válidas: JobName (el nombre del trabajo de AWS Glue), JobRunId (el ID de JobRun o ALL), Type (recuento) y ObservAbilityGroup (error) Estadísticas válidas: SUMA Unidad: recuento	error

Métrica	Descripción	Categoría
glue.error. [categoría de error]	Categoría métrica: error En realidad, se trata de un conjunto de métricas que se actualizan únicamente cuando se produce un error en la ejecución de un trabajo. La categorización de los errores ayuda a clasificar y depurar errores. Cuando se produce un error al ejecutar un trabajo, se clasifica el error que lo ha provocado y la métrica de la categoría de error correspondiente se define en 1. Esto ayuda a realizar un análisis de los errores a lo largo del tiempo, así como un análisis de los errores de todos los trabajos, para identificar las categoría s de errores más comunes y empezar a abordarlas. AWS Glue tiene 28 categorías de error, incluidas las categoría s de error OUT_OF_ME MORY (controlador y ejecutor) , PERMISSION, SYNTAX y THROTTLING. Las categoría s de error también incluyen las categorías COMPILATION, LAUNCH y TIMEOUT. Dimensiones válidas: JobName (el nombre del	error

Métrica	Descripción	Categoría
	trabajo de AWS Glue), JobRunId (el ID de JobRun o ALL), Type (recuento) y ObservAbilityGroup (error) Estadísticas válidas: SUMA Unidad: recuento	
glue.driver.workerUtilization	Categoría de métrica: resource_utilization El porcentaje de los trabajado res asignados que se utilizan realmente. Si no es bueno, el escalado automático puede ayudar. Dimensiones válidas: JobName (el nombre del trabajo de AWS Glue), JobRunId (el ID de JobRun o ALL), Type (calibre), y ObservAbilityGroup (resource _utilization) Estadísticas válidas: promedio, máximo, mínimo, percentilo Unidad: porcentaje	resource_utilization

Métrica	Descripción	Categoría
glue.driver.memory.heap. [disponible | usado]	Categoría de métrica: resource_utilization La memoria de pila disponibl e o utilizada por el controlad or durante la ejecución del trabajo. Esto ayuda a entender las tendencias de uso de la memoria, especialmente a lo largo del tiempo, lo que puede ayudar a evitar posibles fallos, además de depurar los errores relacionados con la memoria. Dimensiones válidas: JobName (el nombre del trabajo de AWS Glue), JobRunId (el ID de JobRun o ALL), Type (calibre), y ObservAbilityGroup (resource _utilization) Estadísticas válidas: Average Unidades: bytes	resource_utilization

Métrica	Descripción	Categoría
glue.driver.memory.heap.used.percentage	Categoría de métrica: resource_utilization El controlador utilizó (%) la memoria acumulada durante la ejecución del trabajo. Esto ayuda a entender las tendencias de uso de la memoria, especialmente a lo largo del tiempo, lo que puede ayudar a evitar posibles fallos, además de depurar los errores relacionados con la memoria. Dimensiones válidas: JobName (el nombre del trabajo de AWS Glue), JobRunId (el ID de JobRun o ALL), Type (calibre), y ObservAbilityGroup (resource_utilization) Estadísticas válidas: Average Unidad: porcentaje	resource_utilization

Métrica	Descripción	Categoría
glue.driver.memory.non-heap.[disponible | usado]	Categoría de métrica: resource_utilization La memoria no acumulada disponible o utilizada por el controlador durante la ejecución del trabajo. Esto ayuda a entender las tendencias de uso de la memoria, especialmente a lo largo del tiempo, lo que puede ayudar a evitar posibles fallos, además de depurar los errores relacionados con la memoria. Dimensiones válidas: JobName (el nombre del trabajo de AWS Glue), JobRunId (el ID de JobRun o ALL), Type (calibre), y ObservAbilityGroup (resource_utilization) Estadísticas válidas: Average Unidades: bytes	resource_utilization

Métrica	Descripción	Categoría
glue.driver.memory.non-heap .used.percentage	Categoría de métrica: resource_utilization El controlador utilizó (%) memoria no acumulada durante la ejecución del trabajo. Esto ayuda a entender las tendencias de uso de la memoria, especialmente a lo largo del tiempo, lo que puede ayudar a evitar posibles fallos, además de depurar los errores relacionados con la memoria. Dimensiones válidas: JobName (el nombre del trabajo de AWS Glue), JobRunId (el ID de JobRun o ALL), Type (calibre), y ObservAbilityGroup (resource _utilization) Estadísticas válidas: Average Unidad: porcentaje	resource_utilization

Métrica	Descripción	Categoría
glue.driver.memory.total. [disponible | usado]	Categoría de métrica: resource_utilization La memoria total disponibl e o utilizada por el controlad or durante la ejecución del trabajo. Esto ayuda a entender las tendencias de uso de la memoria, especialmente a lo largo del tiempo, lo que puede ayudar a evitar posibles fallos, además de depurar los errores relacionados con la memoria. Dimensiones válidas: JobName (el nombre del trabajo de AWS Glue), JobRunId (el ID de JobRun o ALL), Type (calibre), y ObservAbilityGroup (resource _utilization) Estadísticas válidas: Average Unidades: bytes	resource_utilization

Métrica	Descripción	Categoría
glue.driver.memory.total.used.percentage	Categoría de métrica: resource_utilization El controlador utilizó (%) de la memoria total durante la ejecución del trabajo. Esto ayuda a entender las tendencias de uso de la memoria, especialmente a lo largo del tiempo, lo que puede ayudar a evitar posibles fallos, además de depurar los errores relacionados con la memoria. Dimensiones válidas: JobName (el nombre del trabajo de AWS Glue), JobRunId (el ID de JobRun o ALL), Type (calibre), y ObservAbilityGroup (resource_utilization) Estadísticas válidas: Average Unidad: porcentaje	resource_utilization

Métrica	Descripción	Categoría
Glue.all.memory.heap. [disponible | usado]	Categoría de métrica: resource_utilization La memoria de pila utilizada /disponible de los ejecutore s. ALL significa todos los ejecutores. Dimensiones válidas: JobName (el nombre del trabajo de AWS Glue), JobRunId (el ID de JobRun o ALL), Type (calibre), y ObservAbilityGroup (resource _utilization) Estadísticas válidas: Average Unidades: bytes	resource_utilization

Métrica	Descripción	Categoría
glue.ALL.memory.heap.used.percentage	Categoría de métrica: resource_utilization La memoria de pila utilizada por los ejecutores (%). ALL significa todos los ejecutores. Dimensiones válidas: JobName (el nombre del trabajo de AWS Glue), JobRunId (el ID de JobRun o ALL), Type (calibre), y ObservAbilityGroup (resource_utilization) Estadísticas válidas: Average Unidad: porcentaje	resource_utilization

Métrica	Descripción	Categoría
glue.ALL.memory.non-heap.[disponible | usado]	Categoría de métrica: resource_utilization La memoria que no es de pila disponible/utilizada por los ejecutores. ALL significa todos los ejecutores. Dimensiones válidas: JobName (el nombre del trabajo de AWS Glue), JobRunId (el ID de JobRun o ALL), Type (calibre), y ObservAbilityGroup (resource_utilization) Estadísticas válidas: Average Unidades: bytes	resource_utilization

Métrica	Descripción	Categoría
glue.ALL.memory.non-heap.used.percentage	Categoría de métrica: resource_utilization Los ejecutores utilizaron (%) de memoria que no es de pila. ALL significa todos los ejecutores. Dimensiones válidas: JobName (el nombre del trabajo de AWS Glue), JobRunId (el ID de JobRun o ALL), Type (calibre), y ObservAbilityGroup (resource_utilization) Estadísticas válidas: Average Unidad: porcentaje	resource_utilization

Métrica	Descripción	Categoría
glue.ALL.memory.total. [disponible | usado]	Categoría de métrica: resource_utilization La memoria total utilizada/ disponible de los ejecutore s. ALL significa todos los ejecutores. Dimensiones válidas: JobName (el nombre del trabajo de AWS Glue), JobRunId (el ID de JobRun o ALL), Type (calibre), y ObservAbilityGroup (resource _utilization) Estadísticas válidas: Average Unidades: bytes	resource_utilization

Métrica	Descripción	Categoría
glue.ALL.memory.total.used.percentage	Categoría de métrica: resource_utilization La memoria total (%) utilizada por los ejecutores. ALL significa todos los ejecutores. Dimensiones válidas: JobName (el nombre del trabajo de AWS Glue), JobRunId (el ID de JobRun o ALL), Type (calibre), y ObservAbilityGroup (resource_utilization) Estadísticas válidas: Average Unidad: porcentaje	resource_utilization

Métrica	Descripción	Categoría
glue.driver.disk. [disponible_GB | usado_GB]	Categoría de métrica: resource_utilization El espacio en disco disponible o usado por el controlador durante la ejecución del trabajo. Esto ayuda a entender las tendencias de uso del disco, especialmente a lo largo del tiempo, lo que puede ayudar a evitar posibles fallos, además de depurar los errores relacionados con la falta de espacio en disco. Dimensiones válidas: JobName (el nombre del trabajo de AWS Glue), JobRunId (el ID de JobRun o ALL), Type (calibre), y ObservAbilityGroup (resource_utilization) Estadísticas válidas: Average Unidad: Gigabytes	resource_utilization

Métrica	Descripción	Categoría
glue.driver.disk.used.percentage]	Categoría de métrica: resource_utilization El espacio en disco disponible o usado por el controlador durante la ejecución del trabajo. Esto ayuda a entender las tendencias de uso del disco, especialmente a lo largo del tiempo, lo que puede ayudar a evitar posibles fallos, además de depurar los errores relacionados con la falta de espacio en disco. Dimensiones válidas: JobName (el nombre del trabajo de AWS Glue), JobRunId (el ID de JobRun o ALL), Type (calibre), y ObservAbilityGroup (resource_utilization) Estadísticas válidas: Average Unidad: porcentaje	resource_utilization

Métrica	Descripción	Categoría
glue.ALL.disk. [disponible_GB | usado_GB]	Categoría de métrica: resource_utilization El espacio en disco usado/disponible de los ejecutores. ALL significa todos los ejecutores. Dimensiones válidas: JobName (el nombre del trabajo de AWS Glue), JobRunId (el ID de JobRun o ALL), Type (calibre), y ObservAbilityGroup (resource_utilization) Estadísticas válidas: Average Unidad: Gigabytes	resource_utilization

Métrica	Descripción	Categoría
glue.ALL.disk.used.percentage	Categoría de métrica: resource_utilization El espacio en disco disponible/usado/usado (%) de los ejecutores. ALL significa todos los ejecutores. Dimensiones válidas: JobName (el nombre del trabajo de AWS Glue), JobRunId (el ID de JobRun o ALL), Type (calibre), y ObservAbilityGroup (resource_utilization) Estadísticas válidas: Average Unidad: porcentaje	resource_utilization

Métrica	Descripción	Categoría
glue.driver.bytesRead	Categoría métrica: rendimiento El número de bytes leídos por fuente de entrada en esta ejecución de trabajo, y en TODAS las fuentes. Esto ayuda a entender el volumen de datos y sus cambios a lo largo del tiempo, lo que ayuda a tratar problemas como la asimetría de los datos. Dimensiones válidas: JobName (el nombre del trabajo de AWS Glue), JobRunID (el JobRun ID. o ALL), Type (calibre), ObservabilityGroup (resource_utilization) y Source (ubicación de los datos de origen) Estadísticas válidas: Average Unidades: bytes	rendimiento

Métrica	Descripción	Categoría
glue.driver. [Registros leídos | Archivos leídos]	Categoría métrica: rendimiento El número de registros/archivos leídos por fuente de entrada en esta ejecución de trabajo, y en TODAS las fuentes. Esto ayuda a entender el volumen de datos y sus cambios a lo largo del tiempo, lo que ayuda a tratar problemas como la asimetría de los datos. Dimensiones válidas: JobName (el nombre del trabajo de AWS Glue), JobRunID (el JobRun ID. o ALL), Type (calibre), ObservabilityGroup (resource_utilization) y Source (ubicación de los datos de origen) Estadísticas válidas: Average Unidad: recuento	rendimiento

Métrica	Descripción	Categoría
glue.driver.partitionsRead	Categoría métrica: rendimiento El número de particiones leídas por fuente de entrada de Amazon S3 en esta ejecución de trabajo, y para TODAS las fuentes. Dimensiones válidas: JobName (el nombre del trabajo de AWS Glue), JobRunID (el JobRun ID. o ALL), Type (calibre), ObservabilityGroup (resource_utilization) y Source (ubicación de los datos de origen) Estadísticas válidas: Average Unidad: recuento	rendimiento

Métrica	Descripción	Categoría
glue.driver.bytesWrittten	Categoría métrica: rendimiento El número de bytes escritos por receptor de salida en esta ejecución de trabajo, y en TODOS los receptores. Esto ayuda a entender el volumen de datos y cómo evoluciona con el tiempo, lo que ayuda a tratar problemas como la asimetría del procesamiento. Dimensiones válidas: JobName (el nombre del trabajo de AWS Glue), JobRunID (el JobRun ID. o ALL), Type (calibre), ObservabilityGroup (resource_utilization) y Sink (ubicación de los datos de sink) Estadísticas válidas: Average Unidades: bytes	rendimiento

Métrica	Descripción	Categoría
glue.driver. [Registros escritos | Archivos escritos]	Categoría métrica: rendimiento El número de registros o archivos escritos por receptor de salida en esta ejecución de trabajo, y en TODOS los receptores. Esto ayuda a entender el volumen de datos y cómo evoluciona con el tiempo, lo que ayuda a tratar problemas como la asimetría del procesamiento. Dimensiones válidas: JobName (el nombre del trabajo de AWS Glue), JobRunID (el JobRun ID. o ALL), Type (calibre), ObservabilityGroup (resource_utilization) y Sink (ubicación de los datos de sink) Estadísticas válidas: Average Unidad: recuento	rendimiento

Categorías de errores

Categorías de errores	Descripción
COMPILATION_ERROR	Los errores surgen durante la compilación del código de Scala.

Categorías de errores	Descripción
CONNECTION_ERROR	Los errores se producen durante la conexión a un servicio remoto/servicio de host/base de datos, etc.
DISK_NO_SPACE_ERROR	Los errores se producen cuando no queda espacio en el disco del controlador/ejecutor.
OUT_OF_MEMORY_ERROR	Los errores se producen cuando no queda espacio en la memoria del controlador/ejecutor.
IMPORT_ERROR	Los errores se producen al importar dependenc ias.
INVALID_ARGUMENT_ERROR	Los errores se producen cuando los argumento s de entrada son inválidos o ilegales.
PERMISSION_ERROR	Los errores se producen cuando no se tiene el permiso para acceder al servicio, los datos, etc.
RESOURCE_NOT_FOUND_ERROR	Los errores se producen cuando los datos, la ubicación, etc. no existen.
QUERY_ERROR	Los errores se producen por la ejecución de una consulta SQL de Spark.
SYNTAX_ERROR	Los errores se producen cuando hay un error de sintaxis en el script.
THROTTLING_ERROR	Los errores se producen cuando se alcanza el límite de simultaneidad del servicio o se supera el límite de la cuota de servicio.
DATA_LAKE_FRAMEWORK_ERROR	Los errores se deben a un marco de lago de datos con soporte nativo de AWS Glue, como Hudi, Iceberg, etc.

Categorías de errores	Descripción
UNSUPPORTED_OPERATION_ERROR	Los errores se producen al realizar una operación no compatible.
RESOURCES_ALREADY_EXISTS_ERROR	Los errores se producen cuando un recurso que se va a crear o añadir ya existe.
GLUE_INTERNAL_SERVICE_ERROR	Los errores se producen cuando hay un problema de servicio interno de AWS Glue.
GLUE_OPERATION_TIMEOUT_ERROR	Los errores se producen cuando se agota el tiempo de espera de una operación de AWS Glue.
GLUE_VALIDATION_ERROR	Los errores se producen cuando no se ha podido validar un valor requerido para el trabajo de AWS Glue.
GLUE_JOB_BOOKMARK_VERSION_MISMATCH_ERROR	Los errores se producen cuando ejecuta el mismo trabajo en el mismo bucket de origen y escribe en el mismo destino o en uno diferente simultáneamente (concurrencia >1)
LAUNCH_ERROR	Los errores se producen durante la fase de inicio del trabajo de AWS Glue.
DYNAMODB_ERROR	Los errores genéricos se deben al servicio de Amazon DynamoDB.
GLUE_ERROR	Los errores genéricos se deben al servicio de AWS Glue.
LAKEFORMATION_ERROR	Los errores genéricos se deben al servicio de AWS Lake Formation.
REDSHIFT_ERROR	Los errores genéricos se deben al servicio de Amazon Redshift.

Categorías de errores	Descripción
S3_ERROR	Los errores genéricos se deben al servicio de Amazon S3.
SYSTEM_EXIT_ERROR	Error genérico de salida del sistema.
TIMEOUT_ERROR	Los errores genéricos se producen cuando el trabajo falla debido al tiempo de espera de la operación.
UNCLASSIFIED_SPARK_ERROR	Los errores genéricos se deben a Spark.
UNCLASSIFIED_ERROR	Categoría de errores predeterminada.

Limitaciones

> **Note**
>
> `glueContext` debe inicializarse para publicar las métricas.

En la dimensión de origen, el valor es la ruta de Amazon S3 o el nombre de la tabla, según el tipo de fuente. Además, si la fuente es JDBC y se usa la opción de consulta, la cadena de consulta se establece en la dimensión de origen. Si el valor tiene más de 500 caracteres, se recorta hasta 500 caracteres. El valor tiene las siguientes limitaciones:

- Se eliminarán los caracteres que no sean ASCII.
- Si el nombre de la fuente no contiene ningún carácter ASCII, se convierte en <non-ASCII input>.

Limitaciones y consideraciones de las métricas de rendimiento

- Se admiten DataFrame y DynamicFrame basado en DataFrame (por ejemplo, JDBC, lectura desde parquet en Amazon S3); sin embargo, no se admite DynamicFrame basado en RDD (por ejemplo, leer csv, json en Amazon S3, etc.). Técnicamente, se admiten todas las lecturas y escrituras visibles en la interfaz de usuario de Spark.

- La métrica `recordsRead` se emitirá si el origen de datos es una tabla de catálogo y el formato es JSON, CSV, texto o Iceberg.
- Las métricas `glue.driver.throughput.recordsWritten`, `glue.driver.throughput.bytesWritten` y `glue.driver.throughput.filesWritten` no están disponibles en las tablas JDBC e Iceberg.
- Es posible que las métricas se retrasen. Si el trabajo finaliza en aproximadamente un minuto, es posible que no haya métricas de rendimiento en Amazon CloudWatch Metrics.

Monitorización y depuración de trabajo

Puede recopilar métricas acerca de los trabajos de AWS Glue y visualizarlas en las consolas de AWS Glue y Amazon CloudWatch para identificar y solucionar problemas. La generación de perfiles en sus trabajos de AWS Glue requiere los siguientes pasos:

1. Habilitar las métricas:

   a. Habilite la opción Job metrics (Métricas del trabajo) en la definición de trabajo. Puede habilitar la creación de perfiles en la consola de AWS Glue o como parámetro del trabajo. Para obtener más información, consulte Definición de propiedades de trabajo para trabajos de Spark o Parámetros de los trabajos de AWS Glue.

   b. Habilite la opción Métricas de observabilidad AWS Glue en la definición de trabajo. Puede habilitar la observabilidad en la consola de AWS Glue o como parámetro del trabajo. Para obtener más información, consulte Monitorización con métricas de observabilidad de AWS Glue.

2. Confirme que el script de trabajo inicializa `GlueContext`. Por ejemplo, el siguiente fragmento de script inicializa `GlueContext` y muestra en qué parte del script se sitúa el código con perfil. Este formato general se usa en las situaciones de depuración siguientes.

```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job
import time

## @params: [JOB_NAME]
```

```
args = getResolvedOptions(sys.argv, ['JOB_NAME'])

sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args['JOB_NAME'], args)

...
...
code-to-profile
...
...


job.commit()
```

3. Ejecute el trabajo.
4. Visualización de las métricas:
   a. Visualice las métricas de trabajo en la consola AWS Glue e identifique las métricas que están fuera de lo normal para el controlador o un ejecutor.
   b. Compruebe las métricas de observabilidad en la página de monitorización de las ejecuciones de tareas, en la página de detalles de las ejecuciones de tareas o en Amazon CloudWatch. Para obtener más información, consulte Monitorización con métricas de observabilidad de AWS Glue.
5. Acote la causa raíz mediante la métrica identificada.
6. Opcionalmente, confirme la causa raíz mediante el flujo de registros del controlador o el ejecutor de trabajo identificado.

Casos de uso de las métricas de observabilidad de AWS Glue

- Depuración de excepciones de memoria insuficiente e irregularidades relativas al trabajo
- Depuración de etapas exigentes y tareas rezagadas
- Monitoreo del progreso de varios trabajos
- Monitorización de la planificación de la capacidad de DPU
- Uso de la observabilidad de AWS Glue para monitorear el uso de los recursos a fin de reducir los costos

Depuración de excepciones de memoria insuficiente e irregularidades relativas al trabajo

Puede depurar excepciones de memoria insuficiente (OOM) e irregularidades relativas al trabajo en AWS Glue. En las siguientes secciones se describen situaciones de depuración de excepciones de memoria insuficiente del controlador Apache Spark o un ejecutor de Spark.

- Depuración de una excepción de memoria insuficiente del controlador
- Depuración de una excepción de memoria insuficiente del ejecutor

Depuración de una excepción de memoria insuficiente del controlador

En esta situación, un trabajo de Spark está leyendo un gran número de pequeños archivos desde Amazon Simple Storage Service (Amazon S3). Convierte los archivos al formato Apache Parquet y, a continuación, los escribe en Amazon S3. El controlador Spark se está quedando sin memoria. Los datos de Amazon S3 de entrada tienen más de un millón de archivos en diferentes particiones de Amazon S3.

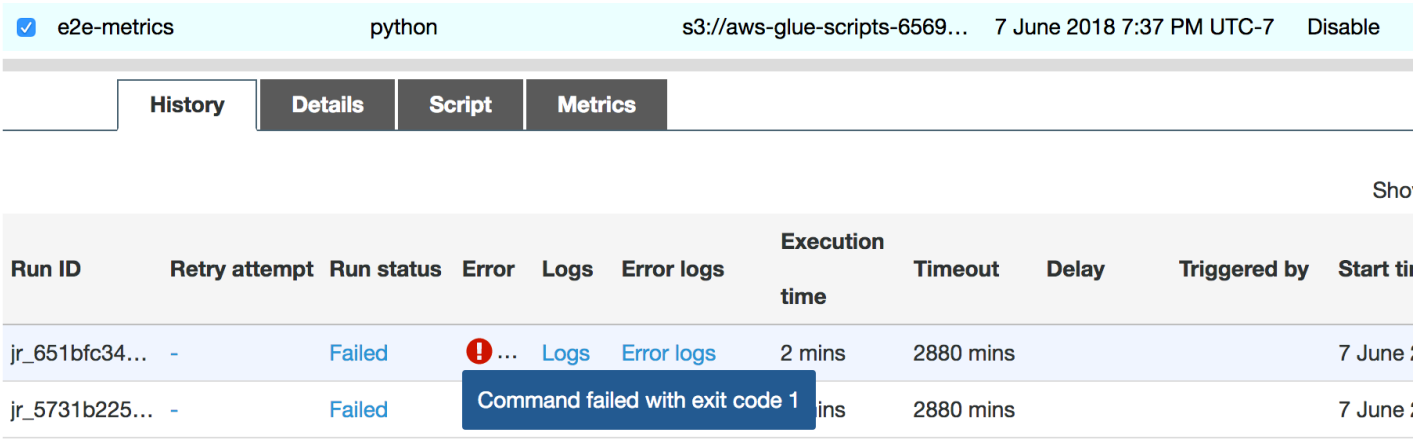
El código con perfil es el siguiente:

```
data = spark.read.format("json").option("inferSchema", False).load("s3://input_path")
data.write.format("parquet").save(output_path)
```

Visualizar las métricas con perfil en la consola de AWS Glue

En el siguiente gráfico se muestra el uso de la memoria como porcentaje del controlador y los ejecutores. Este uso se traza como un punto de datos que se promedia a lo largo de los valores notificados en el último minuto. En el perfil de la memoria del trabajo, puede ver que la memoria del controlador cruza el umbral seguro del 50 % de uso con rapidez. Por otra parte, el uso medio de la memoria en todos los ejecutores sigue siendo inferior al 4 %. Esto es una clara muestra de irregularidad en la ejecución del controlador en el trabajo de Spark.

La ejecución del trabajo produce un error enseguida y aparece el siguiente error en la pestaña History (Historial) de la consola de AWS Glue: Command Failed with Exit Code 1. Esta cadena de error significa que el trabajo falló debido a un error sistémico, que en este caso se trata de la falta de memoria del controlador.



En la consola, elija el enlace Error logs (Registros de error) en la pestaña History (Historial) para confirmar el hallazgo relativo a la falta de memoria del controlador desde CloudWatch Logs. Busque "**`Error`**" en los registros de errores del trabajo para confirmar que se trató, sin duda, de una excepción de memoria insuficiente que imposibilitó la realización del trabajo:

```
# java.lang.OutOfMemoryError: Java heap space
```

```
# -XX:OnOutOfMemoryError="kill -9 %p"
# Executing /bin/sh -c "kill -9 12039"...
```

En la pestaña History (Historial) del trabajo, elija Logs (Registros). Puede encontrar el siguiente rastro de la ejecución del controlador en CloudWatch Logs al empezar el trabajo. El controlador Spark intenta listar todos los archivos en todos los directorios, crea `InMemoryFileIndex` y lanza una tarea por archivo. Esto, a su vez, hace que el controlador Spark tenga que mantener una gran cantidad de estado en memoria para realizar un seguimiento de todas las tareas. Almacena en caché la lista completa de un gran número de archivos para el índice en memoria, lo que da lugar a la falta de memoria de un controlador.

Corregir el procesamiento de varios archivos mediante su agrupación

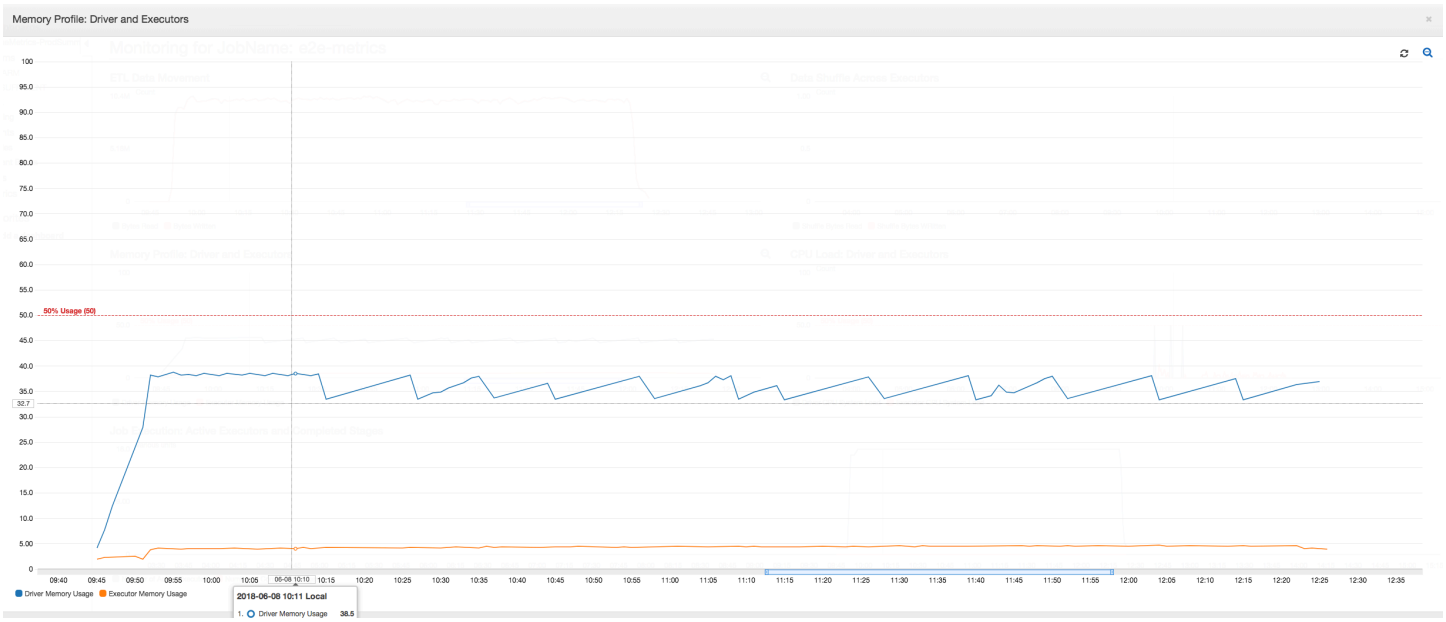
Puede corregir el procesamiento de los numerosos archivos mediante la capacitación de agrupación en AWS Glue. La agrupación se habilita automáticamente al usar marcos dinámicos y cuando el conjunto de datos de entrada tiene un gran número de archivos (más de 50 000). La agrupación le permite fusionar varios archivos en un grupo y deja que una tarea procese todo el grupo en vez de un solo archivo. Como resultado, la cantidad de estado en memoria que almacena el controlador Spark es considerablemente más baja para realizar un seguimiento de menos tareas. Para obtener más información acerca de cómo habilitar manualmente la agrupación de su conjunto de datos, consulte Lectura de archivos de entrada en grupos más grandes.

Para comprobar el perfil de la memoria del trabajo de AWS Glue, genere perfiles en el siguiente código con la agrupación habilitada:

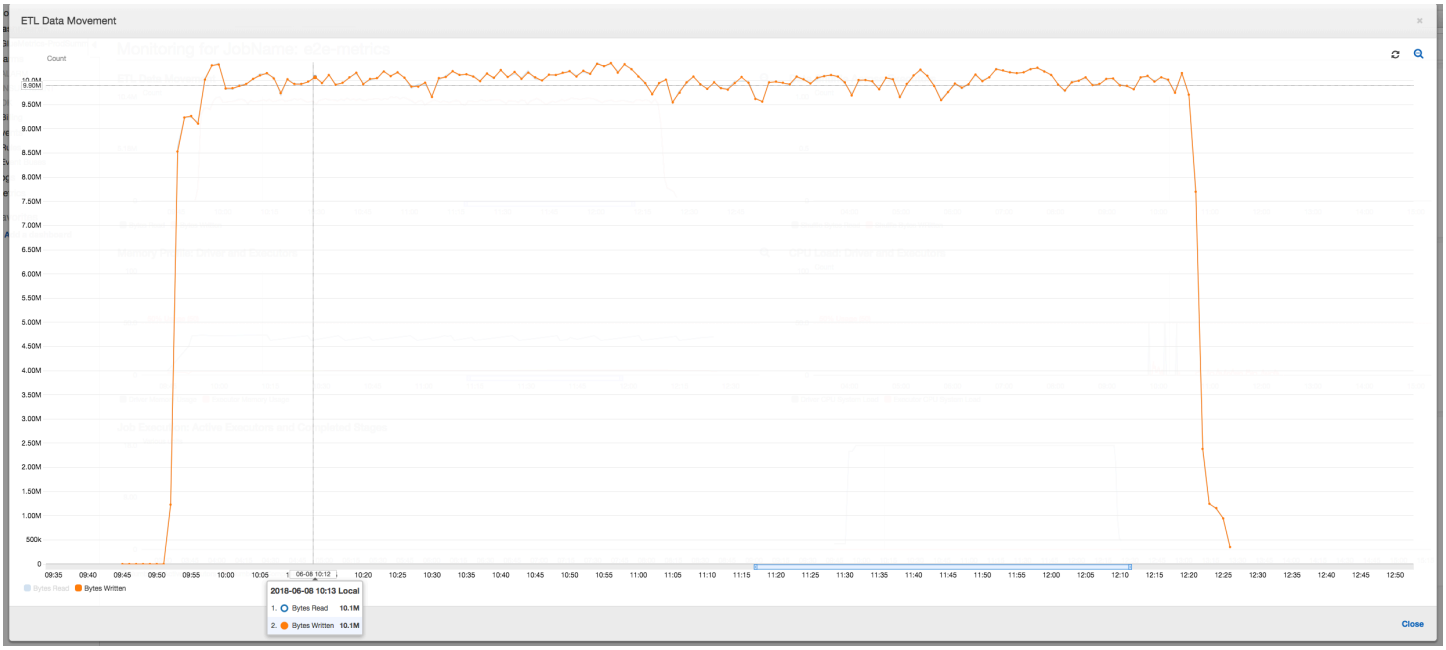
```
df = glueContext.create_dynamic_frame_from_options("s3", {'paths': ["s3://input_path"],
 "recurse":True, 'groupFiles': 'inPartition'}, format="json")
datasink = glueContext.write_dynamic_frame.from_options(frame = df, connection_type
 = "s3", connection_options = {"path": output_path}, format = "parquet",
 transformation_ctx = "datasink")
```

Puede monitorear el perfil de la memoria y el movimiento de datos de ETL en el perfil de trabajo de AWS Glue.

El controlador se ejecuta por debajo del umbral del 50 % de uso de la memoria durante todo el trabajo de AWS Glue. Los ejecutores transmiten los datos desde Amazon S3, los procesan y los escriben en Amazon S3. Como resultado, consumen un porcentaje de memoria inferior al 5 % en cualquier momento.



El perfil de movimiento de datos a continuación muestra el número total de bytes de Amazon S3 que todos los ejecutores leen y escriben en el último minuto a medida que progresa el trabajo. Ambos siguen un patrón similar, puesto que todos los ejecutores transmiten los datos. El trabajo termina de procesar el millón de archivos en menos de tres horas.



Depuración de una excepción de memoria insuficiente del ejecutor

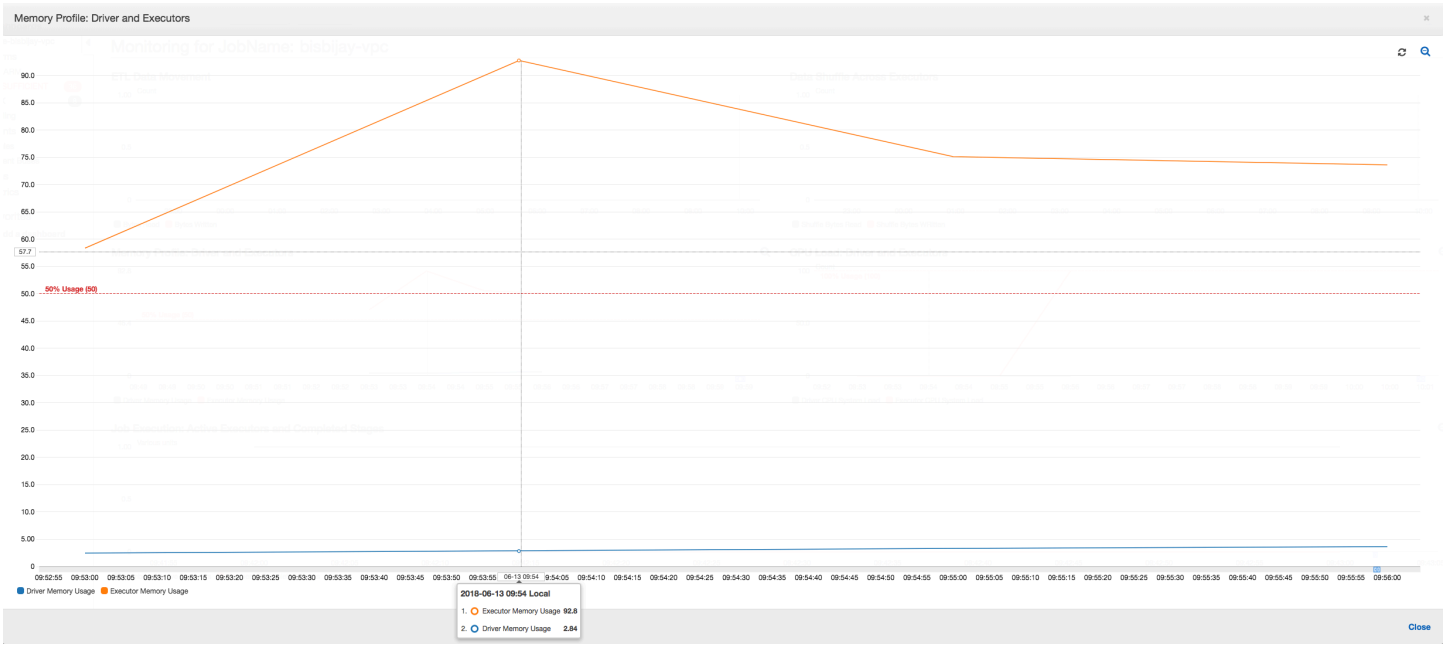
En esta situación, puede aprender a depurar excepciones de memoria insuficiente que podrían producirse en los ejecutores Apache Spark. El siguiente código usa el lector de Spark MySQL para leer una gran tabla de unos 34 millones de filas en un DataFrame de Spark. A continuación, la

escribe en Amazon S3 en formato Parquet. Puede proporcionar las propiedades de la conexión y usar las configuraciones de Spark predeterminadas para leer la tabla.

```
val connectionProperties = new Properties()
connectionProperties.put("user", user)
connectionProperties.put("password", password)
connectionProperties.put("Driver", "com.mysql.jdbc.Driver")
val sparkSession = glueContext.sparkSession
val dfSpark = sparkSession.read.jdbc(url, tableName, connectionProperties)
dfSpark.write.format("parquet").save(output_path)
```

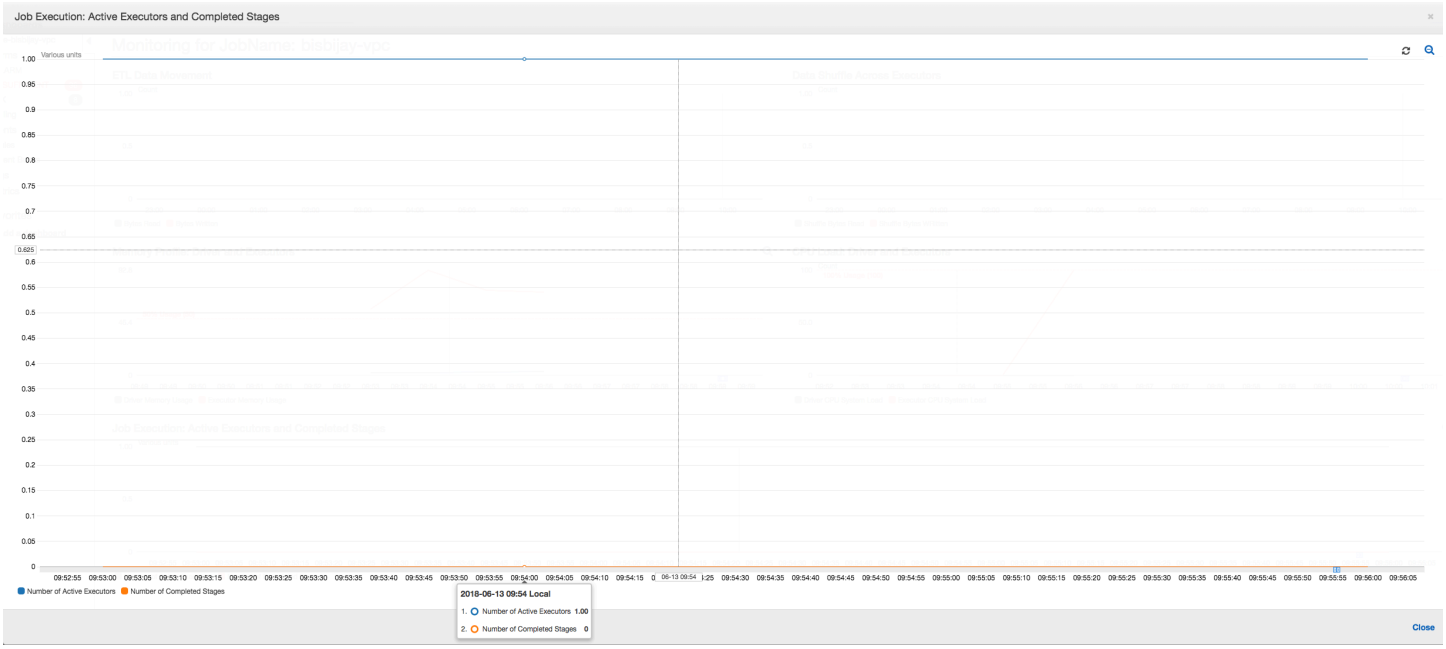
Visualizar las métricas con perfil en la consola de AWS Glue

Si la pendiente del gráfico de uso de memoria es positiva y supera el 50 por ciento, si se produce un error en el trabajo antes de que se emita la siguiente métrica, la causa bien se puede deber al agotamiento de memoria. En el siguiente gráfico se muestra que, transcurrido un minuto desde el inicio de la ejecución, el uso medio de la memoria en todos los ejecutores aumenta rápidamente por encima del 50 %. El uso llega a alcanzar el 92 % y Apache Hadoop YARN cancela el contenedor que ejecuta el ejecutor.

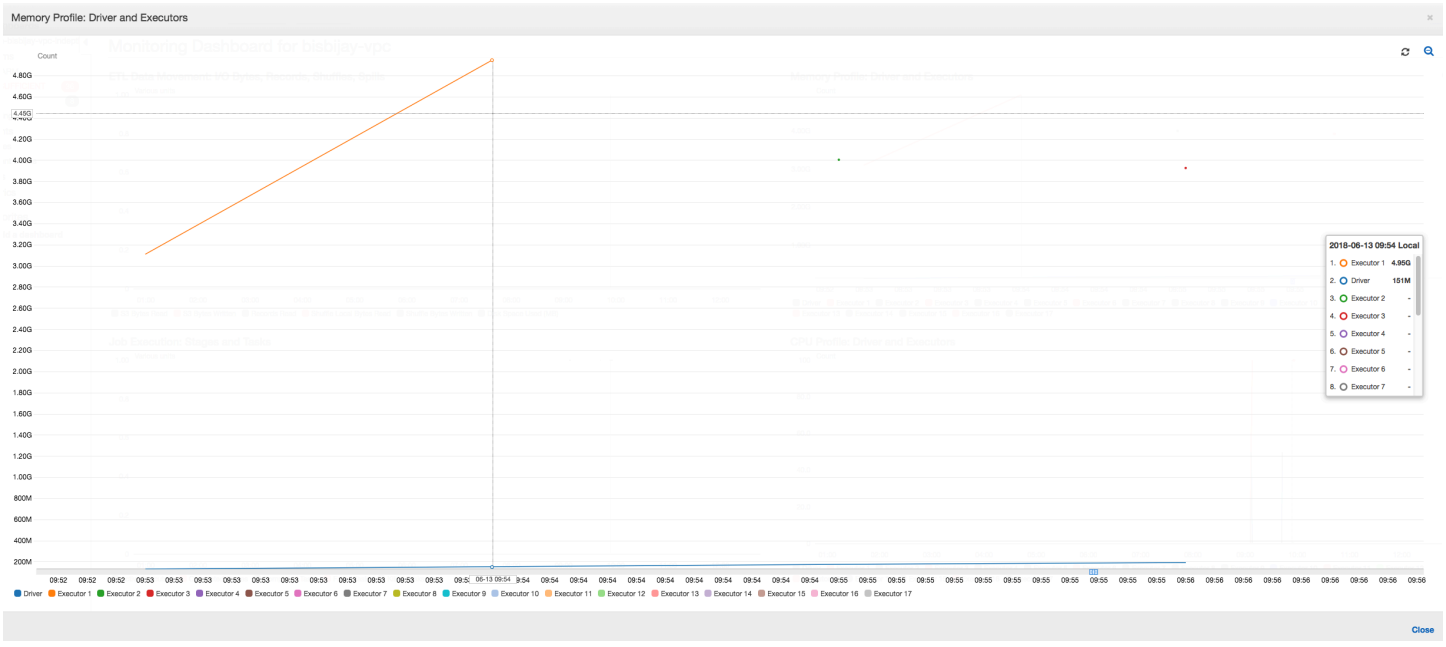


Tal como aparece en el siguiente gráfico, siempre hay un único ejecutor en ejecución hasta que el trabajo produce un error. Esto se debe al lanzamiento de un nuevo ejecutor para reemplazar el ejecutor cancelado. Las lecturas del origen de datos JDBC no se paralelizan de forma

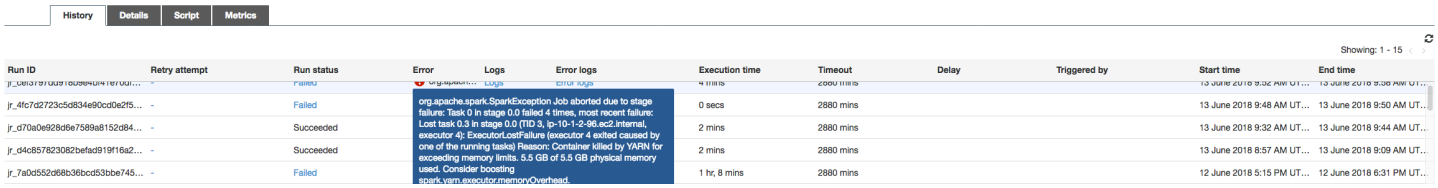
predeterminada, ya que requeriría que se particionara la tabla en una columna y que se abrieran varias conexiones. Como resultado, solo un ejecutor lee la tabla completa de forma secuencial.



Tal como se muestra en el siguiente gráfico, Spark intenta lanzar una nueva tarea cuatro veces antes de producir el trabajo el error. Puede ver el perfil de la memoria de tres ejecutores. Cada ejecutor usa rápidamente toda su memoria. El cuarto ejecutor se queda sin memoria y el trabajo produce un error. Como resultado, su métrica no se notifica inmediatamente.



Puede confirmar desde la cadena de error de la consola de AWS Glue que el trabajo produjo un error debido a excepciones de memoria insuficiente, como se muestra en la siguiente imagen.



Registros de salida del trabajo: para confirmar aún más su hallazgo de una excepción de memoria insuficiente, examine CloudWatch Logs. Al buscar **`Error`**, verá cómo se cancelan los cuatro ejecutores casi al mismo tiempo como se muestra en el panel de métricas. YARN termina todos, ya que superan sus límites de memoria.

Ejecutor 1

```
18/06/13 16:54:29 WARN YarnAllocator: Container killed by YARN for exceeding
 memory limits. 5.5 GB of 5.5 GB physical memory used. Consider boosting
 spark.yarn.executor.memoryOverhead.
18/06/13 16:54:29 WARN YarnSchedulerBackend$YarnSchedulerEndpoint: Container killed
 by YARN for exceeding memory limits. 5.5 GB of 5.5 GB physical memory used. Consider
 boosting spark.yarn.executor.memoryOverhead.
18/06/13 16:54:29 ERROR YarnClusterScheduler: Lost executor 1 on
 ip-10-1-2-175.ec2.internal: Container killed by YARN for exceeding
 memory limits. 5.5 GB of 5.5 GB physical memory used. Consider boosting
 spark.yarn.executor.memoryOverhead.
18/06/13 16:54:29 WARN TaskSetManager: Lost task 0.0 in stage 0.0 (TID 0,
 ip-10-1-2-175.ec2.internal, executor 1): ExecutorLostFailure (executor 1
 exited caused by one of the running tasks) Reason: Container killed by YARN for
 exceeding memory limits. 5.5 GB of 5.5 GB physical memory used. Consider boosting
 spark.yarn.executor.memoryOverhead.
```

Ejecutor 2

```
18/06/13 16:55:35 WARN YarnAllocator: Container killed by YARN for exceeding
 memory limits. 5.8 GB of 5.5 GB physical memory used. Consider boosting
 spark.yarn.executor.memoryOverhead.
18/06/13 16:55:35 WARN YarnSchedulerBackend$YarnSchedulerEndpoint: Container killed
 by YARN for exceeding memory limits. 5.8 GB of 5.5 GB physical memory used. Consider
 boosting spark.yarn.executor.memoryOverhead.
18/06/13 16:55:35 ERROR YarnClusterScheduler: Lost executor 2 on
 ip-10-1-2-16.ec2.internal: Container killed by YARN for exceeding
 memory limits. 5.8 GB of 5.5 GB physical memory used. Consider boosting
 spark.yarn.executor.memoryOverhead.
18/06/13 16:55:35 WARN TaskSetManager: Lost task 0.1 in stage 0.0 (TID 1,
 ip-10-1-2-16.ec2.internal, executor 2): ExecutorLostFailure (executor 2 exited
```

```
 caused by one of the running tasks) Reason: Container killed by YARN for
 exceeding memory limits. 5.8 GB of 5.5 GB physical memory used. Consider boosting
 spark.yarn.executor.memoryOverhead.
```

Ejecutor 3

```
18/06/13 16:56:37 WARN YarnAllocator: Container killed by YARN for exceeding
 memory limits. 5.8 GB of 5.5 GB physical memory used. Consider boosting
 spark.yarn.executor.memoryOverhead.
18/06/13 16:56:37 WARN YarnSchedulerBackend$YarnSchedulerEndpoint: Container killed
 by YARN for exceeding memory limits. 5.8 GB of 5.5 GB physical memory used. Consider
 boosting spark.yarn.executor.memoryOverhead.
18/06/13 16:56:37 ERROR YarnClusterScheduler: Lost executor 3 on
 ip-10-1-2-189.ec2.internal: Container killed by YARN for exceeding
 memory limits. 5.8 GB of 5.5 GB physical memory used. Consider boosting
 spark.yarn.executor.memoryOverhead.
18/06/13 16:56:37 WARN TaskSetManager: Lost task 0.2 in stage 0.0 (TID 2,
 ip-10-1-2-189.ec2.internal, executor 3): ExecutorLostFailure (executor 3
 exited caused by one of the running tasks) Reason: Container killed by YARN for
 exceeding memory limits. 5.8 GB of 5.5 GB physical memory used. Consider boosting
 spark.yarn.executor.memoryOverhead.
```

Ejecutor 4

```
18/06/13 16:57:18 WARN YarnAllocator: Container killed by YARN for exceeding
 memory limits. 5.5 GB of 5.5 GB physical memory used. Consider boosting
 spark.yarn.executor.memoryOverhead.
18/06/13 16:57:18 WARN YarnSchedulerBackend$YarnSchedulerEndpoint: Container killed
 by YARN for exceeding memory limits. 5.5 GB of 5.5 GB physical memory used. Consider
 boosting spark.yarn.executor.memoryOverhead.
18/06/13 16:57:18 ERROR YarnClusterScheduler: Lost executor 4 on
 ip-10-1-2-96.ec2.internal: Container killed by YARN for exceeding
 memory limits. 5.5 GB of 5.5 GB physical memory used. Consider boosting
 spark.yarn.executor.memoryOverhead.
18/06/13 16:57:18 WARN TaskSetManager: Lost task 0.3 in stage 0.0 (TID 3,
 ip-10-1-2-96.ec2.internal, executor 4): ExecutorLostFailure (executor 4 exited
 caused by one of the running tasks) Reason: Container killed by YARN for
 exceeding memory limits. 5.5 GB of 5.5 GB physical memory used. Consider boosting
 spark.yarn.executor.memoryOverhead.
```

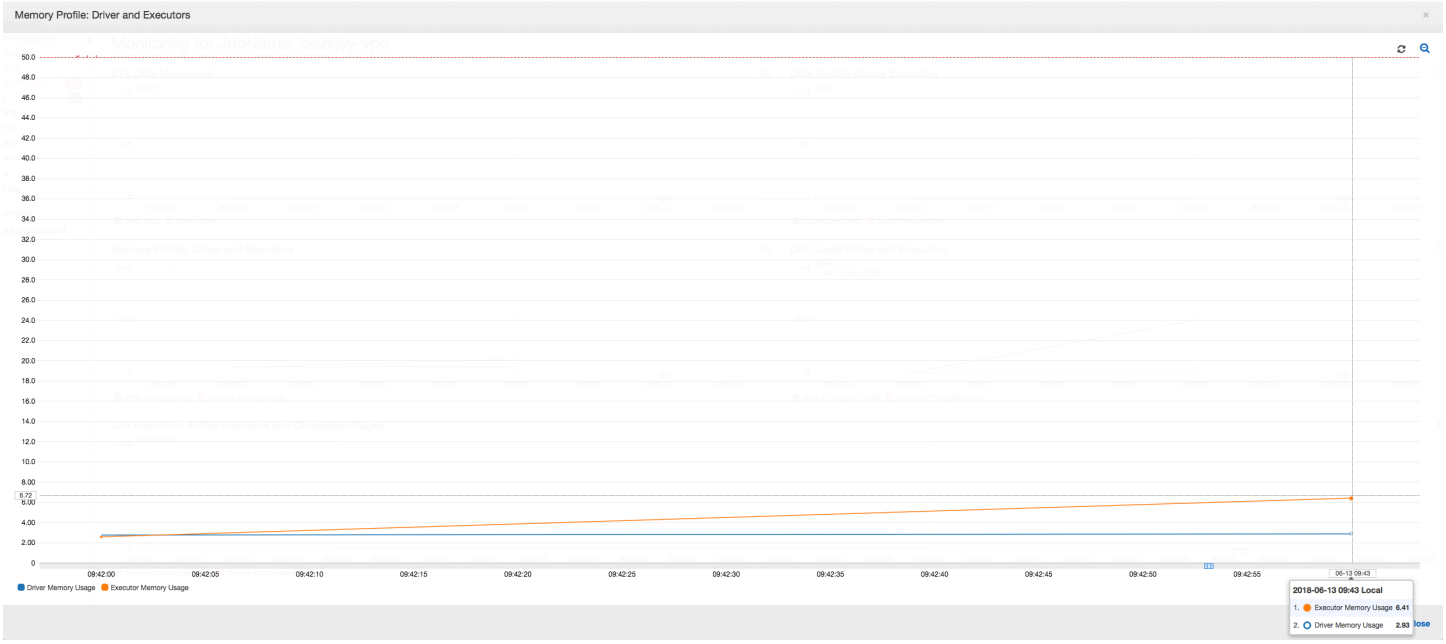
Corregir la configuración del tamaño de búsqueda mediante marcos dinámicos de AWS Glue

El ejecutor se quedó sin memoria durante la lectura de la tabla de JDBC, ya que la configuración predeterminada del tamaño de búsqueda de JDBC de Spark es cero. Esto significa que el controlador JDBC del ejecutor de Spark intenta recuperar los 34 millones de filas de la base de datos de forma conjunta y almacenarlas en caché, aunque Spark se transmita a través de las filas de una en una. Con Spark, puede evitar esta situación estableciendo el parámetro de tamaño de búsqueda en un valor predeterminado distinto de cero.

También puede solucionar este problema con marcos dinámicos de AWS Glue en su lugar. De forma predeterminada, los marcos dinámicos utilizan un tamaño de captura de 1000 filas, que suele ser un valor suficiente. Como resultado, el ejecutor no usa más del 7 % de su memoria total. El trabajo de AWS Glue finaliza en menos de dos minutos con un único ejecutor solamente. Aunque el enfoque recomendado es utilizar los marcos dinámicos de AWS Glue, también se puede definir el tamaño de captura con la propiedad `fetchsize` de Apache Spark. Consulte la guía de Spark SQL, DataFrames y Datasets.

```
val (url, database, tableName) = {
 ("jdbc_url", "db_name", "table_name")
 }
val source = glueContext.getSource(format, sourceJson)
val df = source.getDynamicFrame
glueContext.write_dynamic_frame.from_options(frame = df, connection_type = "s3",
 connection_options = {"path": output_path}, format = "parquet", transformation_ctx =
 "datasink")
```

Métricas con perfil normales: la memoria del ejecutor con marcos dinámicos de AWS Glue nunca supera el umbral seguro, como se muestra en la siguiente imagen. Se transmite en las filas de la base de datos y almacena en caché solo 1000 filas en el controlador JDBC en cualquier momento. No se producen excepciones de memoria insuficiente.



Depuración de etapas exigentes y tareas rezagadas

Puede usar perfiles de trabajo de AWS Glue para identificar etapas exigentes y tareas rezagadas en sus trabajos de extracción, transformación y carga (ETL). Una tarea rezagada tarda mucho más que las demás tareas en una etapa de un trabajo de AWS Glue. Como resultado, la etapa tarda más en completarse, lo que también retrasa el tiempo de ejecución total del trabajo.

Fusión de archivos de entrada pequeños en archivos de salida más grandes

Una tarea rezagada puede producirse si hay una distribución no uniforme de trabajo entre las diferentes tareas, o un sesgo de datos da lugar a un mayor procesamiento de datos por parte de una tarea.

Puede generar perfiles en el siguiente código, un patrón común en Apache Spark, para fusionar un gran número de archivos pequeños en archivos de salida más grandes. En este ejemplo, el conjunto de datos de entrada tiene 32 GB de archivos comprimidos Gzip JSON. El conjunto de datos de salida tiene casi 190 GB de archivos JSON sin comprimir.

El código con perfil es el siguiente:

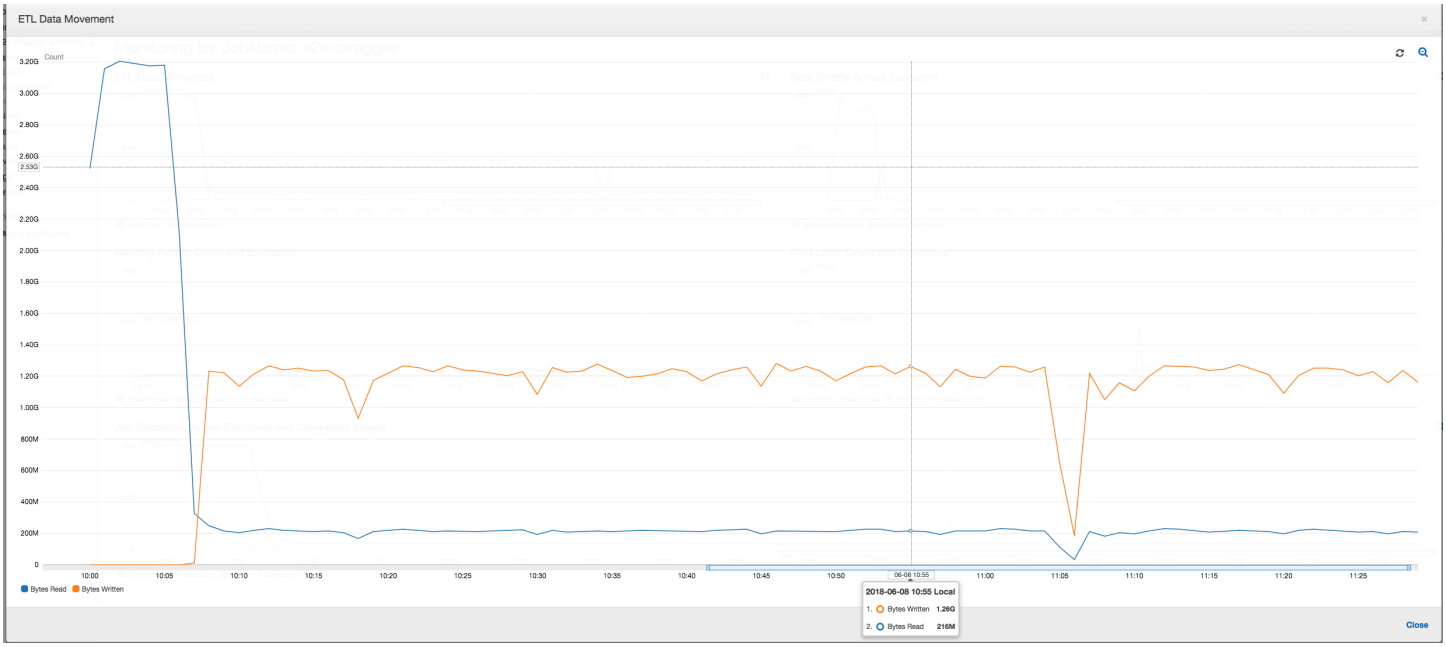
```
datasource0 = spark.read.format("json").load("s3://input_path")
df = datasource0.coalesce(1)
df.write.format("json").save(output_path)
```

Visualizar las métricas con perfil en la consola de AWS Glue

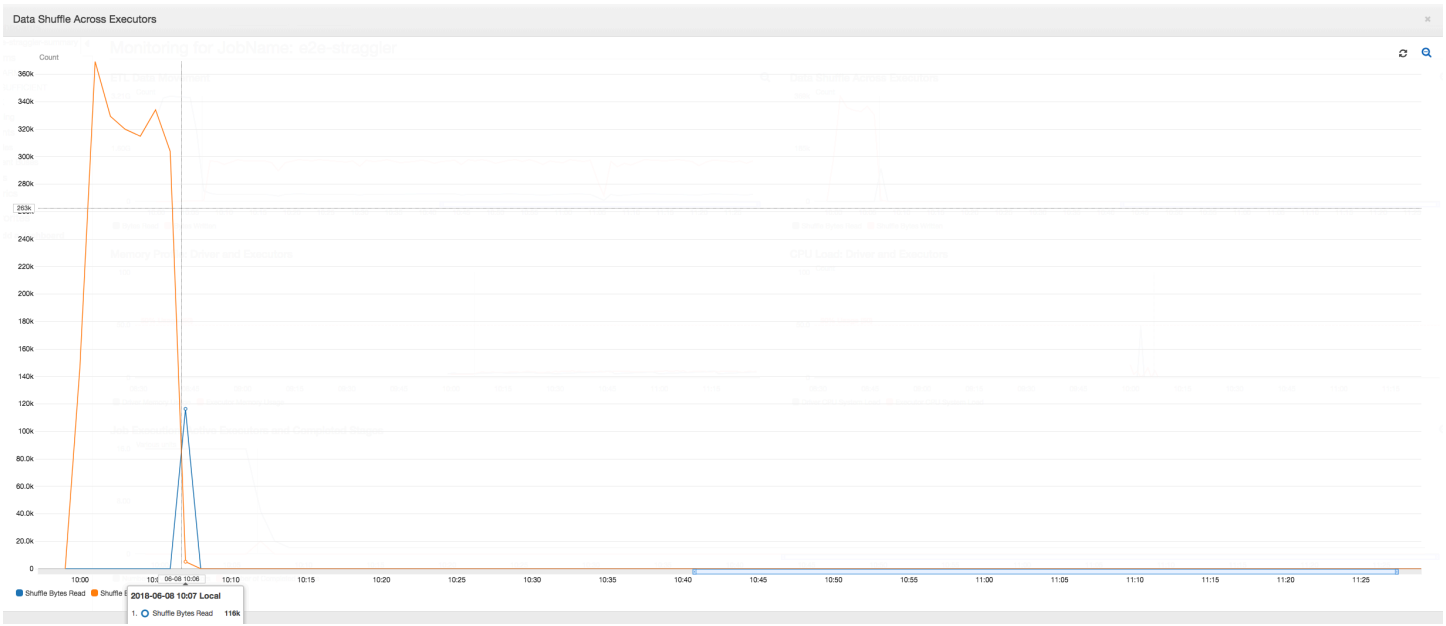
Puede generar perfiles en su trabajo para examinar cuatro conjuntos de métricas diferentes:

- Movimiento de datos de ETL
- Mezcla de datos entre los ejecutores
- Ejecución de trabajo
- Perfil de la memoria

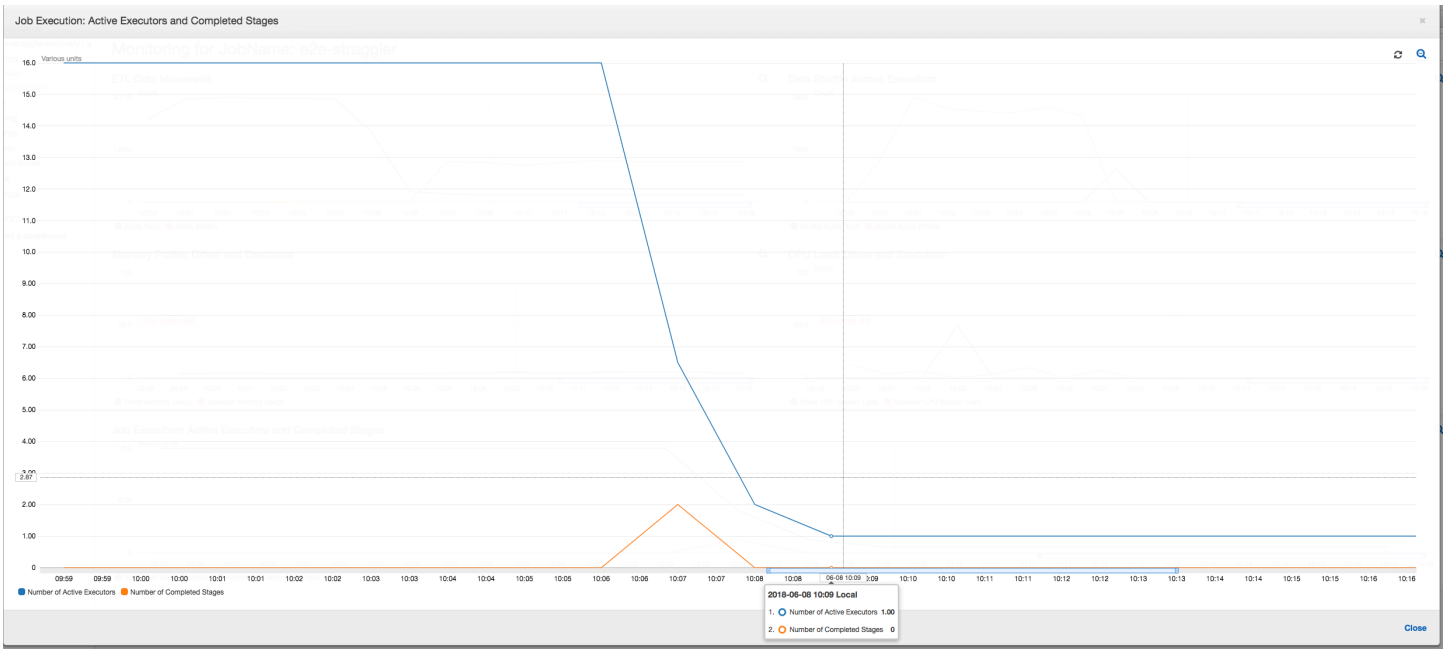
Movimiento de datos de ETL: en el perfil ETL Data Movement (Movimiento de datos de ETL), todos los ejecutores leen los bytes con bastante rapidez en la primera etapa que se completa en los primeros seis minutos. Sin embargo, el tiempo de ejecución del trabajo total es de aproximadamente una hora, componiéndose principalmente de las escrituras de datos.



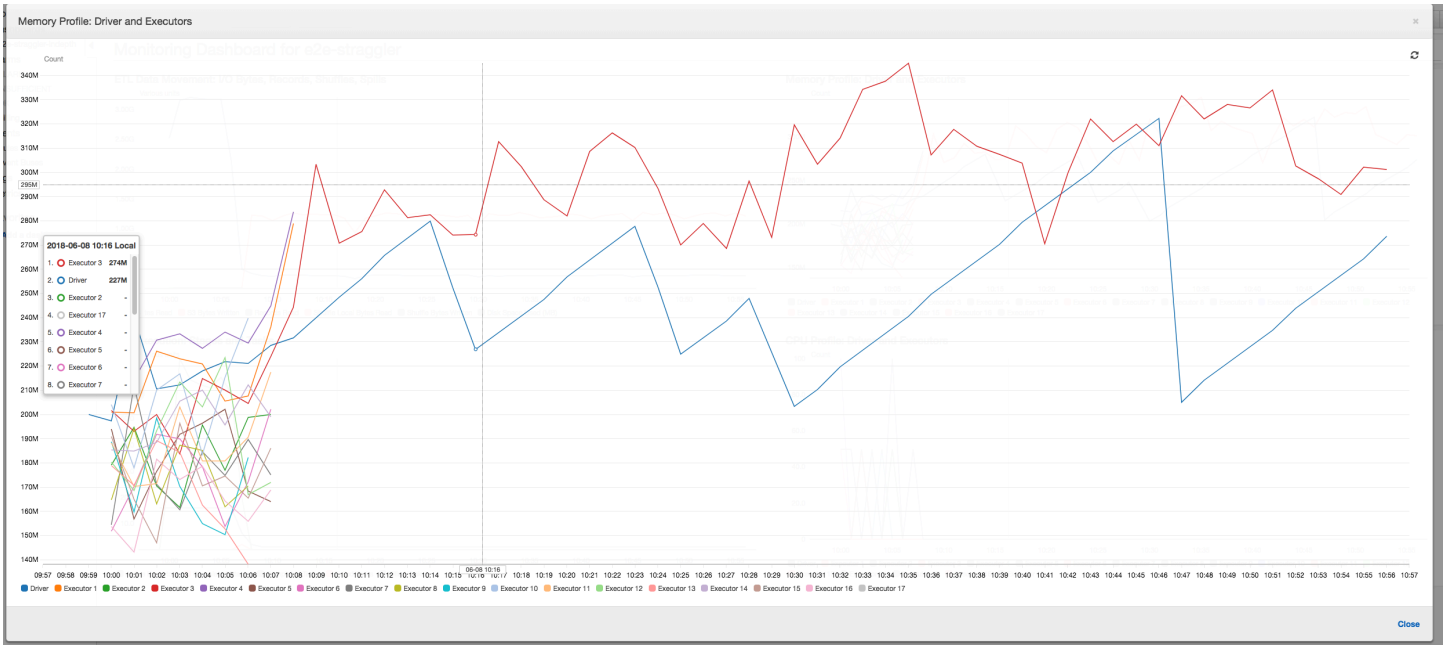
Mezcla de datos entre los ejecutores: el número de bytes leídos y escritos durante la mezcla también muestra un pico antes de finalizar la segunda etapa, como indican las métricas Job Execution (Ejecución de trabajo) y Data Shuffle (Mezcla de datos). Una vez que se mezclan los datos de todos los ejecutores, las lecturas y escrituras siguen solo a partir del tercer ejecutor.



Ejecución de trabajo: tal como se muestra en el gráfico a continuación, todos los demás ejecutores están inactivos y finalmente se renuncia a ellos por el tiempo 10:09. En este momento, el número total de ejecutores disminuye a solo uno. Esto muestra claramente que el tercer ejecutor se compone de la tarea rezagada que más tiempo de ejecución emplea y que contribuye a la mayor parte del tiempo de ejecución del trabajo.



Perfil de la memoria: después de las primeras dos etapas, solo el tercer ejecutor consume memoria de forma activa para procesar los datos. Los demás ejecutores están simplemente inactivos o se ha renunciado a ellos poco después de completarse las primeras dos etapas.



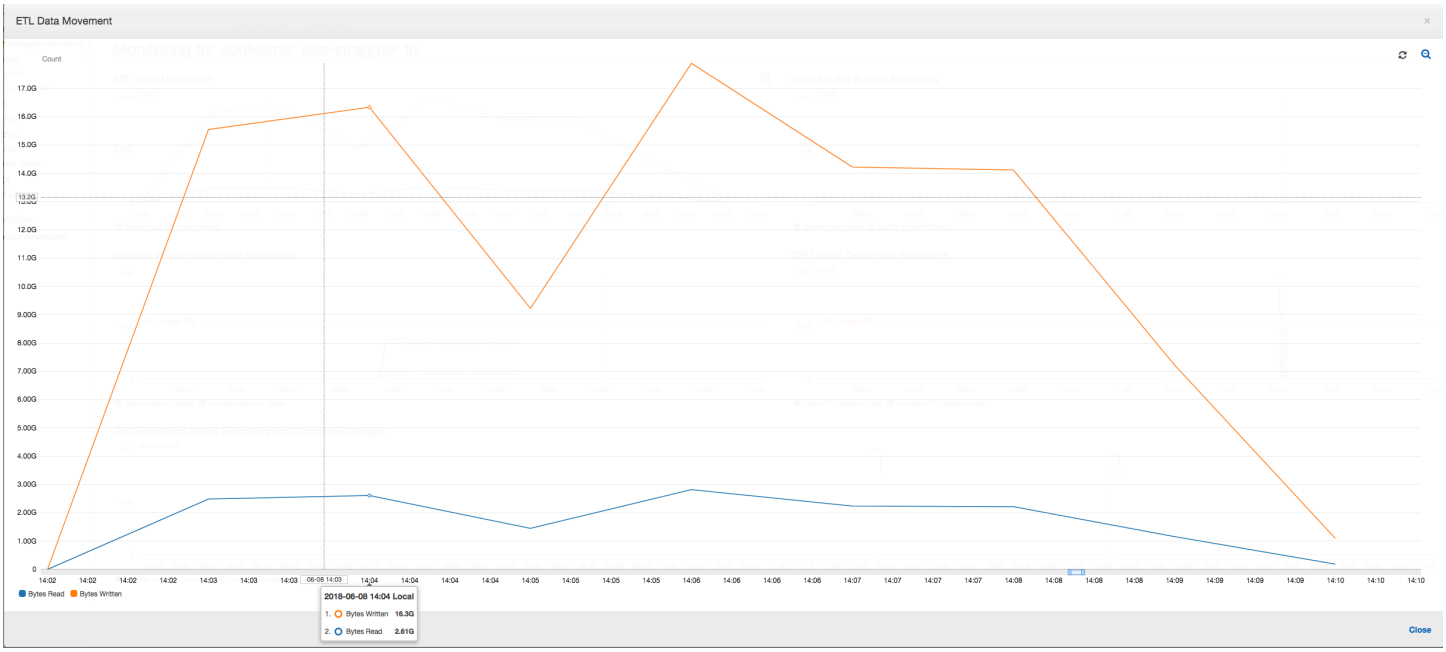
Corregir ejecutores rezagados mediante agrupación

Puede evitar los ejecutores rezagados mediante la capacitación de agrupación en AWS Glue. Utilice la agrupación para distribuir los datos de manera uniforme entre todos los ejecutores y fusionar archivos en archivos más grandes mediante el uso de todos los ejecutores disponibles en el clúster. Para obtener más información, consulte Lectura de archivos de entrada en grupos más grandes.

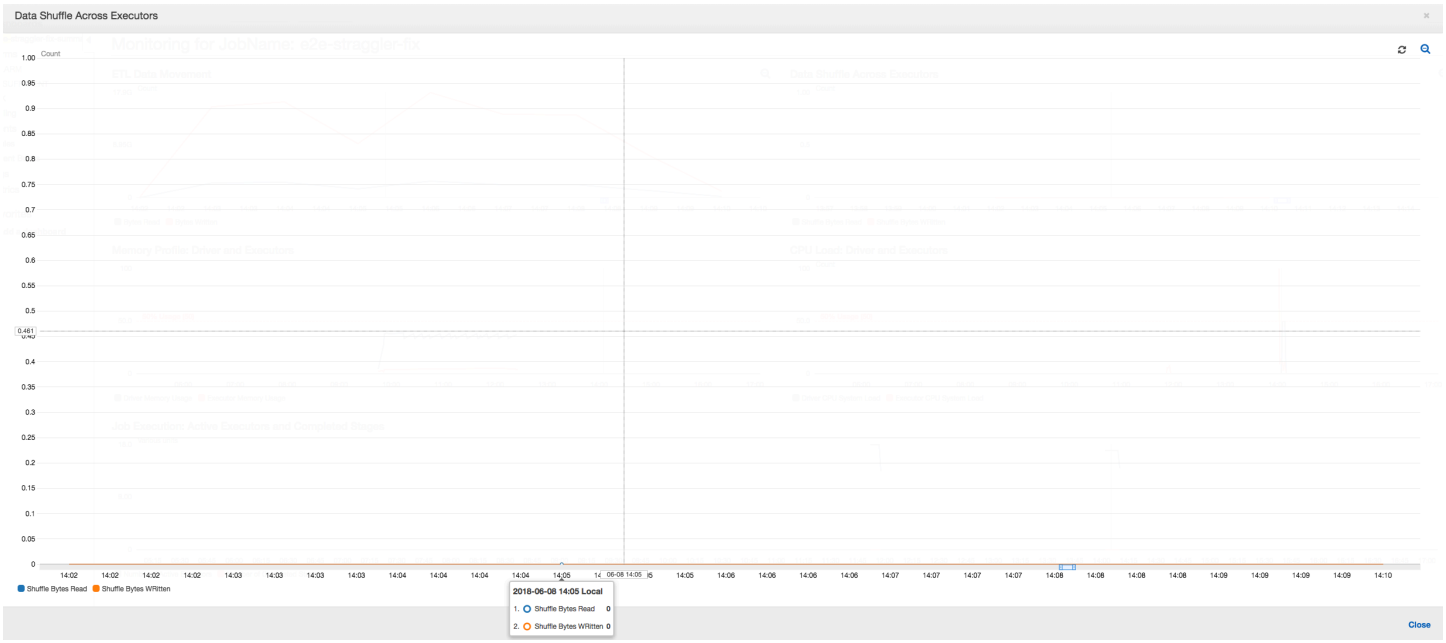
Para comprobar los movimientos de datos de ETL en el trabajo de AWS Glue, genere perfiles en el siguiente código con la agrupación habilitada:

```
df = glueContext.create_dynamic_frame_from_options("s3", {'paths': ["s3://input_path"],
 "recurse":True, 'groupFiles': 'inPartition'}, format="json")
datasink = glueContext.write_dynamic_frame.from_options(frame = df, connection_type =
 "s3", connection_options = {"path": output_path}, format = "json", transformation_ctx
 = "datasink4")
```

Movimiento de datos de ETL: las escrituras de datos ahora se transmiten en paralelo con las lecturas de datos a lo largo del tiempo de ejecución del trabajo. Como resultado, el trabajo finaliza en ocho minutos, mucho más rápido que antes.

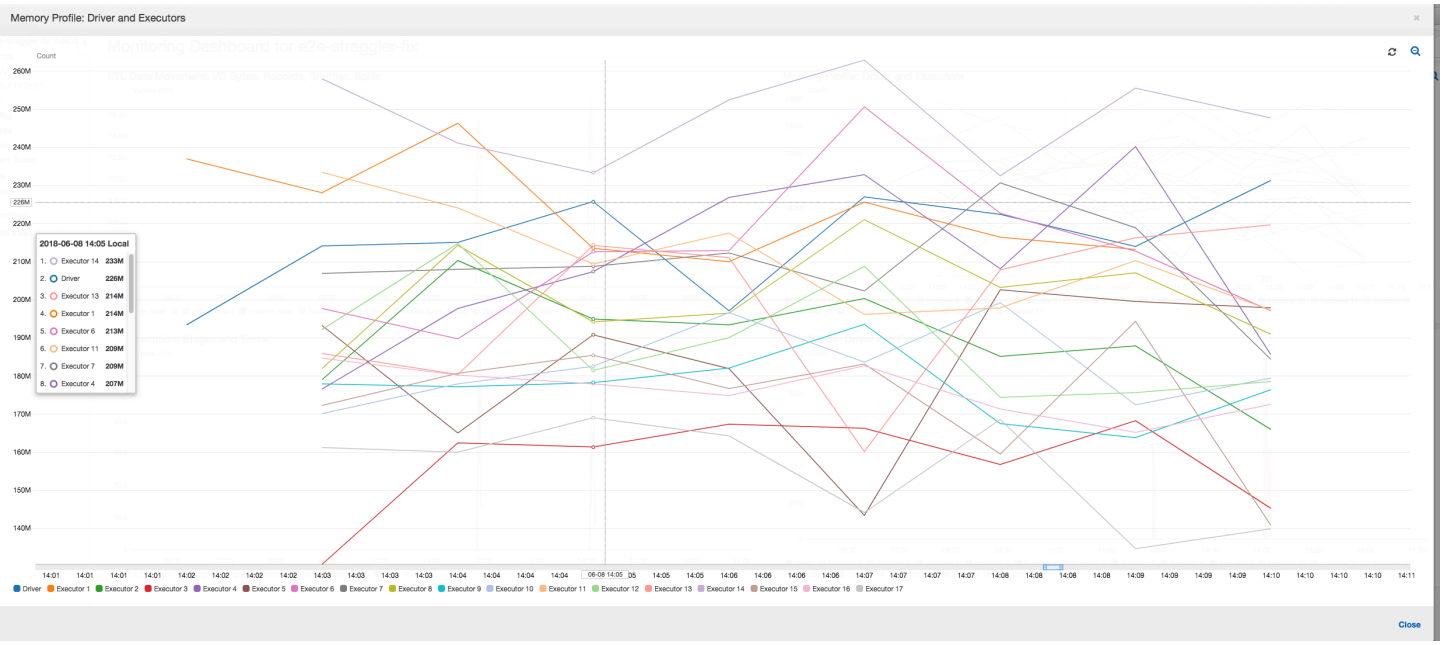


Mezcla de datos entre los ejecutores: como los archivos de entrada se fusionan durante las lecturas mediante la capacitación de agrupación, no hay ninguna mezcla de datos costosa después de las lecturas de datos.



Ejecución de trabajo: las métricas de ejecución de trabajo muestran que el número total de ejecutores activos que ejecutan y procesan datos permanecen bastante constantes. No hay ningún rezagado en el trabajo. Todos los ejecutores están activos y no se renuncia a ellos hasta completarse el trabajo. Como no hay mezcla de datos intermedia alguna entre los ejecutores, solo hay una única etapa en el trabajo.

Perfil de memoria: las métricas muestran el consumo de memoria activa entre todos los ejecutores, lo que confirma una vez más que hay actividad en todos los ejecutores. Como los datos se reciben y escriben en paralelo, la huella de memoria total de todos los ejecutores es más o menos uniforme y está muy por debajo del umbral seguro de todos los ejecutores.



Monitoreo del progreso de varios trabajos

Puede generar perfiles en varios trabajos de AWS Glue de forma conjunta y monitorear el flujo de datos entre ellos. Se trata de un patrón de flujo de trabajo habitual y requiere el monitoreo

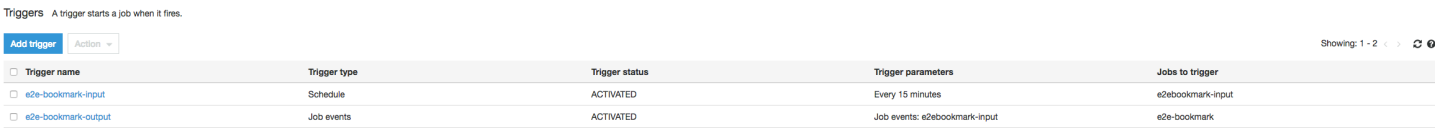
del progreso de trabajos individuales, las tareas pendientes de procesamiento de datos, el reprocesamiento de datos y los marcadores de trabajos.

Temas

- Código con perfil
- Visualizar las métricas con perfil en la consola de AWS Glue
- Corregir el procesamiento de los archivos

Código con perfil

En este flujo de trabajo, tiene dos trabajos: un trabajo de entrada y un trabajo de salida. El trabajo de entrada está programado para ejecutarse cada 30 minutos mediante un disparador periódico. El trabajo de salida está programado para ejecutarse tras cada ejecución correcta del trabajo de entrada. Estos trabajos programados se controlan mediante disparadores del trabajo.



Trabajo de entrada: este trabajo lee datos desde una ubicación de Amazon Simple Storage Service (Amazon S3), los transforma mediante `ApplyMapping` y los escribe en una ubicación de Amazon S3 provisional. El siguiente código es código con perfil para el trabajo de entrada:

```
datasource0 = glueContext.create_dynamic_frame.from_options(connection_type="s3",
 connection_options = {"paths": ["s3://input_path"],
 "useS3ListImplementation":True,"recurse":True}, format="json")
applymapping1 = ApplyMapping.apply(frame = datasource0, mappings = [map_spec])
datasink2 = glueContext.write_dynamic_frame.from_options(frame = applymapping1,
 connection_type = "s3", connection_options = {"path": staging_path, "compression":
 "gzip"}, format = "json")
```

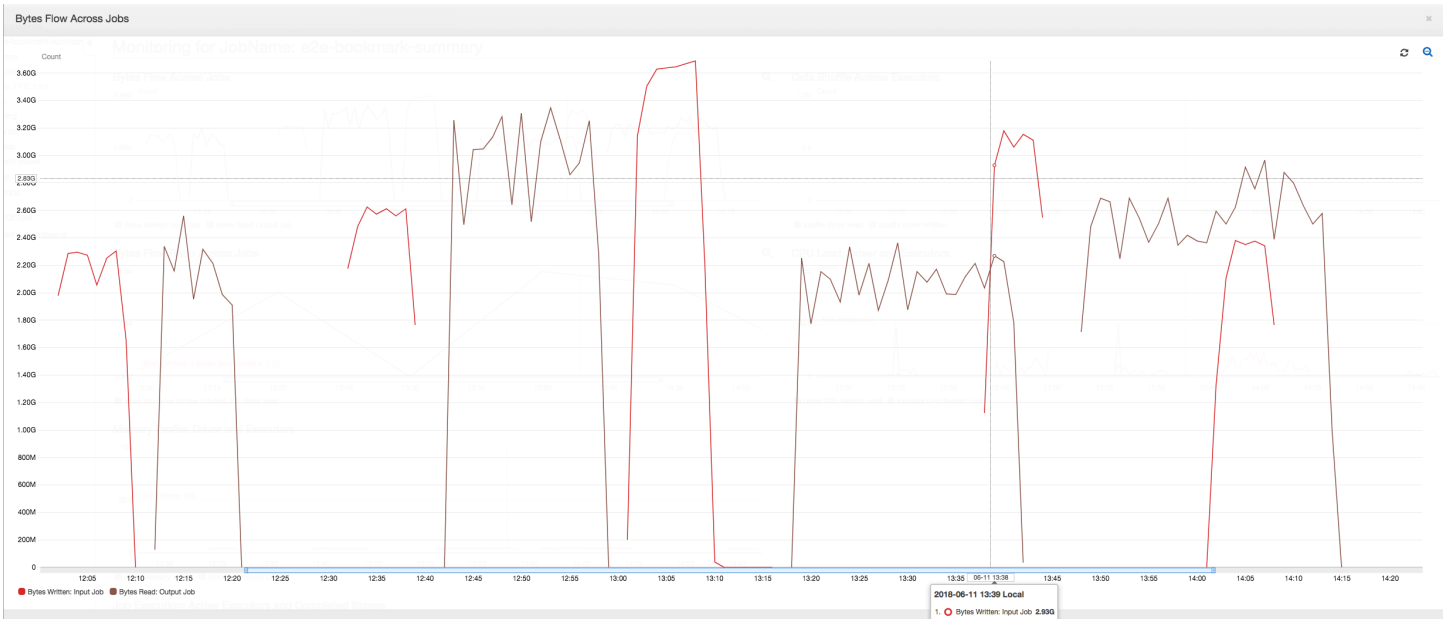
Trabajo de salida: este trabajo lee la salida del trabajo de entrada desde la ubicación provisional en Amazon S3, la vuelve a transformar y la escribe en un destino:

```
datasource0 = glueContext.create_dynamic_frame.from_options(connection_type="s3",
 connection_options = {"paths": [staging_path],
 "useS3ListImplementation":True,"recurse":True}, format="json")
applymapping1 = ApplyMapping.apply(frame = datasource0, mappings = [map_spec])
```

```
datasink2 = glueContext.write_dynamic_frame.from_options(frame = applymapping1,
 connection_type = "s3", connection_options = {"path": output_path}, format = "json")
```

Visualizar las métricas con perfil en la consola de AWS Glue

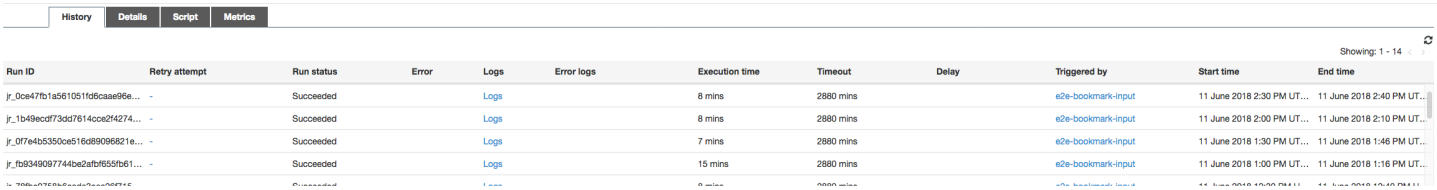
El siguiente panel superpone la métrica escrita de bytes de Amazon S3 desde el trabajo de entrada en la métrica leída de bytes de Amazon S3 en la misma escala de tiempo del trabajo de salida. La escala de tiempo muestra diferentes ejecuciones de trabajo de los trabajos de entrada y salida. El trabajo de entrada (mostrado en rojo) comienza cada 30 minutos. El trabajo de salida (mostrado en marrón) comienza al finalizar el trabajo de entrada, con una simultaneidad máxima de 1.



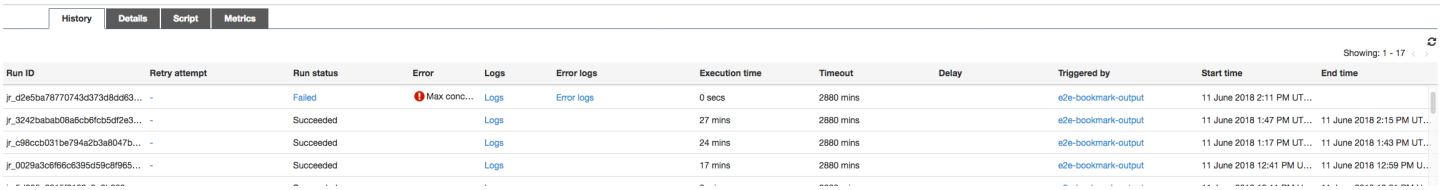
En este ejemplo, los marcadores de trabajos no están habilitados. No se usa ningún contexto de transformación para habilitar marcadores de trabajos en el código de script.

Historial de trabajos: los trabajos de entrada y salida tienen varias ejecuciones, como se muestra en la pestaña History (Historial), las cuales comienzan a partir de las 12:00 h.

El trabajo de entrada de la consola de AWS Glue tiene este aspecto:

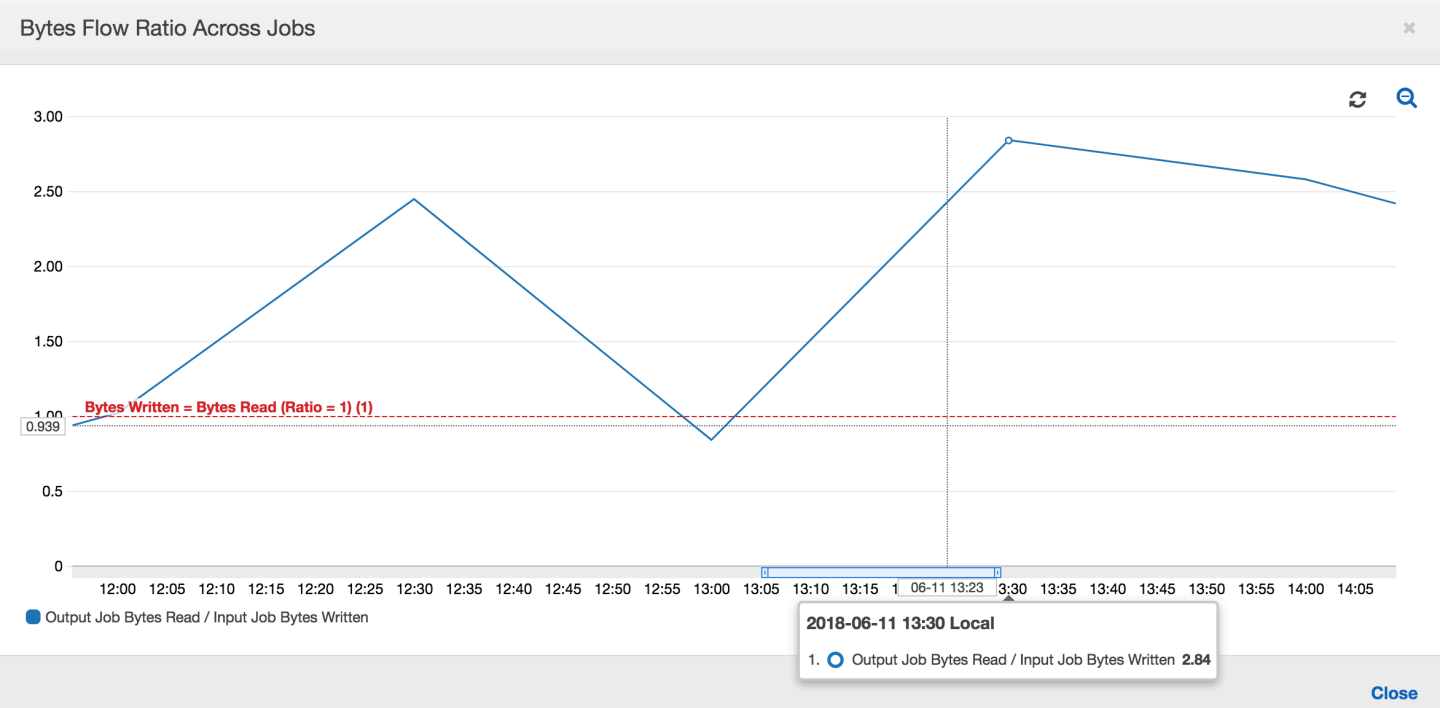


En la siguiente imagen se muestra el trabajo de salida:



Primeras ejecuciones de trabajo: como se muestra en el siguiente gráfico de bytes de datos leídos y escritos, las primeras ejecuciones de trabajo de los trabajos de entrada y salida entre las 12:00 y las 12:30 h muestran más o menos la misma área bajo las curvas. Esas áreas representan los bytes de Amazon S3 escritos por el trabajo de entrada y los bytes de Amazon S3 leídos por el trabajo de salida. Estos datos también están confirmados por la relación de bytes de Amazon S3 escritos (suma de más de 30 minutos, la frecuencia del desencadenador de trabajo para el trabajo de entrada). El punto de datos de la relación para la ejecución de trabajo de entrada que comenzó a las 12:00 h también es 1.

En el siguiente gráfico se muestra la relación de flujo de datos entre todas las ejecuciones de trabajo:



Segundas ejecuciones de trabajo: en la segunda ejecución de trabajo, existe una clara diferencia entre el número de bytes leídos por el trabajo de salida y el número de bytes escritos por el trabajo de entrada (compare el área bajo la curva entre las dos ejecuciones de trabajo para el trabajo de salida, o bien las áreas en la segunda ejecución de los trabajos de entrada y salida). La relación de los bytes leídos y escritos muestra que el trabajo de salida lee alrededor de 2,5 veces los datos escritos por el trabajo de entrada en el segundo periodo de 30 minutos de 12:30 a 13:00 h. Esto se

debe a que el trabajo de salida reprocesó la salida de la primera ejecución de trabajo del trabajo de entrada porque los marcadores de trabajos no estaban habilitados. Una relación superior a 1 muestra que hay tareas pendientes adicionales de datos que el trabajo de salida procesó.

Terceras ejecuciones de trabajo: el trabajo de entrada es bastante coherente en términos del número de bytes escritos (consulte el área bajo las curvas rojas). Sin embargo, la tercera ejecución de trabajo del trabajo de entrada se ejecutó más tiempo del que se había previsto (consulte la larga cola de la curva roja). Como resultado, la tercera ejecución de trabajo del trabajo de salida comenzó tarde. La tercera ejecución de trabajo procesó solo una fracción de los datos acumulados en la ubicación provisional durante los 30 minutos restantes entre las 13:00 y las 13:30 h. La relación del flujo de bytes muestra que solo procesó 0,83 de los datos escritos por la tercera ejecución de trabajo del trabajo de entrada (consulte la relación a las 13:00 h).

Solapamiento de los trabajos de entrada y salida: la cuarta ejecución de trabajo del trabajo de entrada comenzó a las 13:30 h según el programa, antes de finalizar la tercera ejecución de trabajo del trabajo de salida. Existe un solapamiento parcial entre estas dos ejecuciones de trabajo. Sin embargo, la tercera ejecución de trabajo del trabajo de salida captura solo los archivos que listó en la ubicación provisional de Amazon S3 cuando empezó alrededor de las 13:17 h. Esta consta de la salida de todos los datos de las primeras ejecuciones de trabajo del trabajo de entrada. La relación real a las 13:30 h es de aproximadamente 2,75. La tercera ejecución de trabajo del trabajo de salida procesó alrededor de 2,75 veces los datos escritos por la cuarta ejecución de trabajo del trabajo de entrada de 13:00 a 14:00 h.

Tal como muestran estas imágenes, el trabajo de salida reprocesa datos de la ubicación provisional de todas las ejecuciones de trabajo anteriores del trabajo de entrada. Como resultado, la cuarta ejecución de trabajo para el trabajo de salida es la más larga y se solapa con la quinta ejecución de trabajo al completo del trabajo de entrada.

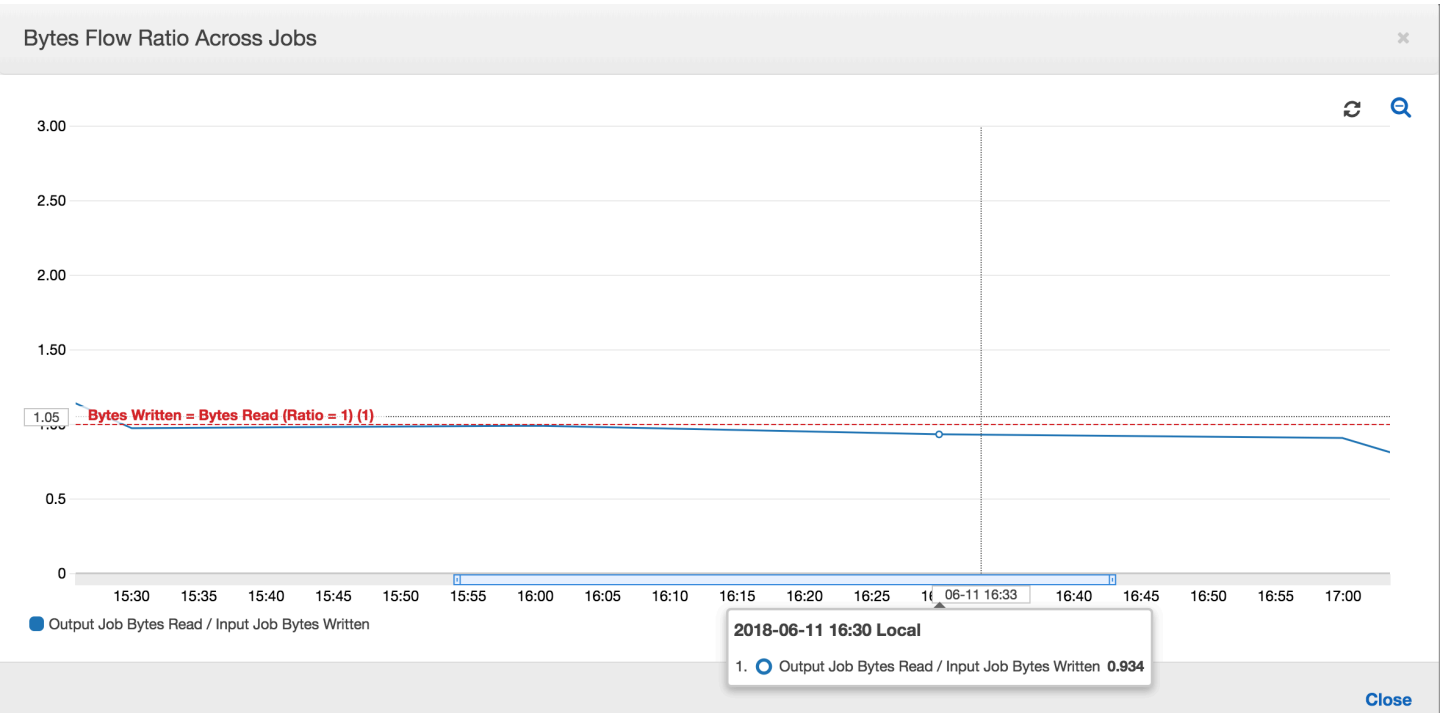
Corregir el procesamiento de los archivos

Debe garantizar que el trabajo de salida procese solo los archivos que no han procesado las ejecuciones de trabajo anteriores del trabajo de salida. Para ello, habilite marcadores de trabajos y establezca el contexto de transformación en el trabajo de salida, como se indica a continuación:

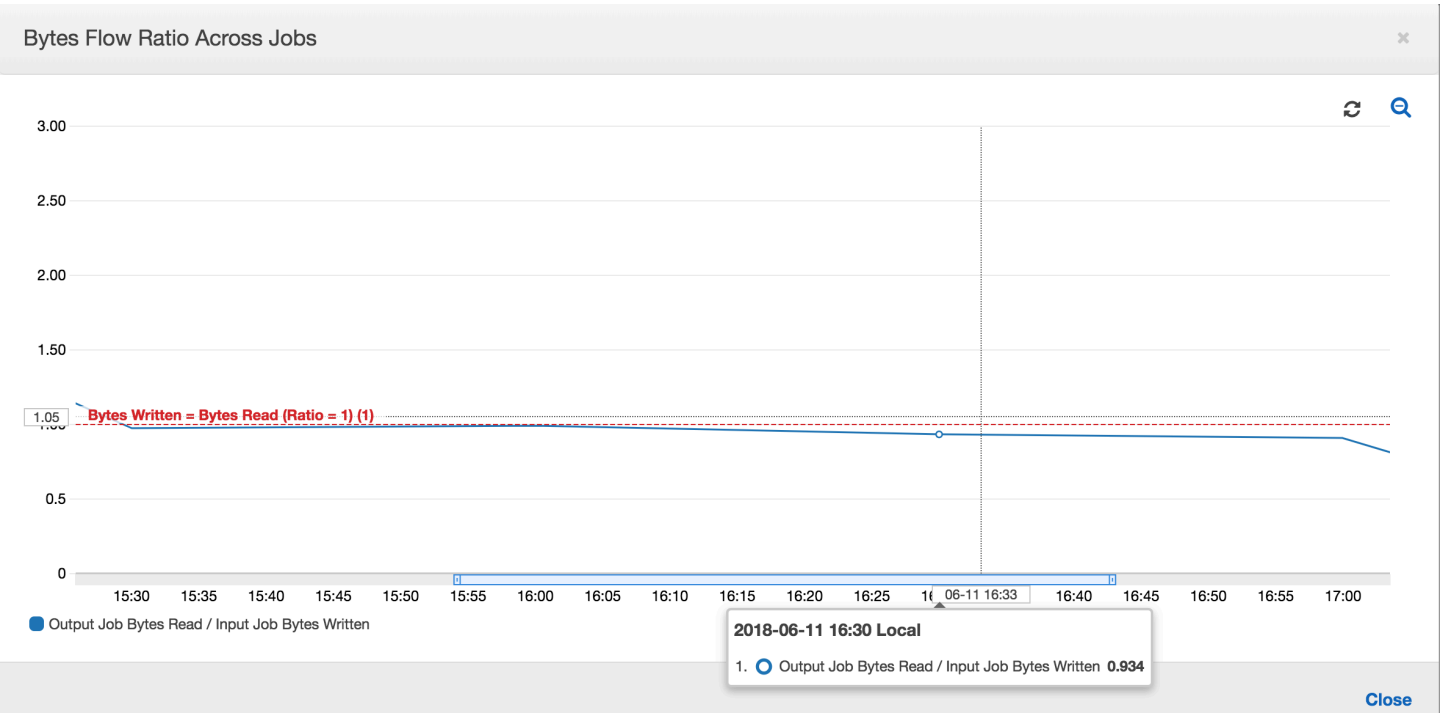
```
datasource0 = glueContext.create_dynamic_frame.from_options(connection_type="s3",
 connection_options = {"paths": [staging_path],
 "useS3ListImplementation":True,"recurse":True}, format="json", transformation_ctx =
 "bookmark_ctx")
```

Con los marcadores de trabajos habilitados, el trabajo de salida no reprocesa los datos en la ubicación provisional de todas las ejecuciones de trabajo anteriores del trabajo de entrada. En la siguiente imagen donde se muestran los datos leídos y escritos, el área bajo la curva marrón es bastante coherente y similar a la de las curvas rojas.

Las relaciones de flujo de bytes también permanecen más o menos próximas a 1 al no haber datos adicionales procesados.



Una ejecución de trabajo para el trabajo de salida comienza y captura los archivos en la ubicación provisional antes de que la siguiente ejecución de trabajo de entrada comience a poner más datos en la ubicación provisional. Siempre que siga haciendo esto, procesará solo los archivos capturados de la ejecución de trabajo de entrada anterior, mientras que la relación permanece próxima a 1.



Supongamos que el trabajo de entrada tarda más de lo esperado y, como resultado, el trabajo de salida captura archivos en la ubicación provisional de dos ejecuciones de trabajo de entrada. A continuación, la relación es mayor que 1 para esa ejecución de trabajo de salida. Sin embargo, las siguientes ejecuciones de trabajo del trabajo de salida no procesan ningún archivo que ya hayan procesado las ejecuciones de trabajo anteriores del trabajo de salida.

Monitorización de la planificación de la capacidad de DPU

Puede usar las métricas de trabajos en AWS Glue para estimar el número de unidades de procesamiento de datos (DPU) que se puede utilizar para escalar en horizontal un trabajo de AWS Glue.

> **Note**
>
> Esta página solo corresponde para versiones 0.9 y 1.0 de AWS Glue. Las versiones posteriores de AWS Glue contienen funciones de ahorro de costos que introducen consideraciones adicionales a la hora de planificar la capacidad.

Temas

- Código con perfil
- Visualizar las métricas con perfil en la consola de AWS Glue
- Determinar la capacidad de DPU óptima

Código con perfil

El siguiente script lee una partición de Amazon Simple Storage Service (Amazon S3) que contiene 428 archivos gzip JSON. El script aplica un mapeo para cambiar los nombres de los campos y convertirlos y escribirlos en Amazon S3 en formato Apache Parquet. Aprovisione 10 DPU según el valor predeterminado y ejecute este trabajo.

```
datasource0 = glueContext.create_dynamic_frame.from_options(connection_type="s3",
 connection_options = {"paths": [input_path],
 "useS3ListImplementation":True,"recurse":True}, format="json")
applymapping1 = ApplyMapping.apply(frame = datasource0, mappings = [(map_spec])
datasink2 = glueContext.write_dynamic_frame.from_options(frame = applymapping1,
 connection_type = "s3", connection_options = {"path": output_path}, format =
 "parquet")
```

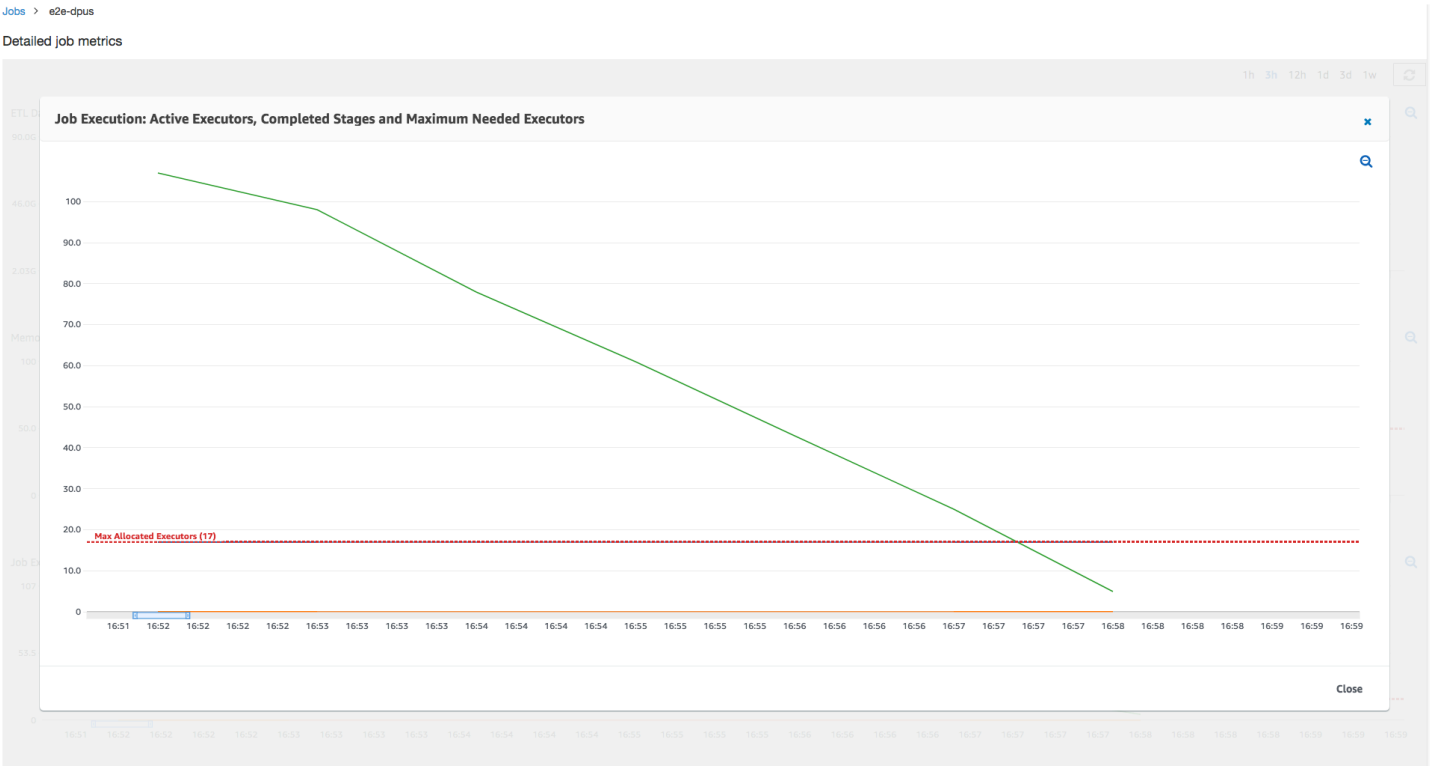
Visualizar las métricas con perfil en la consola de AWS Glue

Ejecución de trabajo 1: en esta ejecución de trabajo, mostramos cómo encontrar si hay DPU de aprovisionamiento deficiente en el clúster. La funcionalidad de ejecución de trabajos de AWS Glue muestra el número de ejecutores que se ejecutan activamente total, el número de fases completadas y el número máximo de ejecutores necesarios.

El número máximo de ejecutores necesarios se calcula añadiendo el número total de tareas en ejecución y tareas pendientes, y dividiéndolo entre las tareas por ejecutor. Este resultado es una medida del número total de ejecutores necesarios para satisfacer la carga actual.

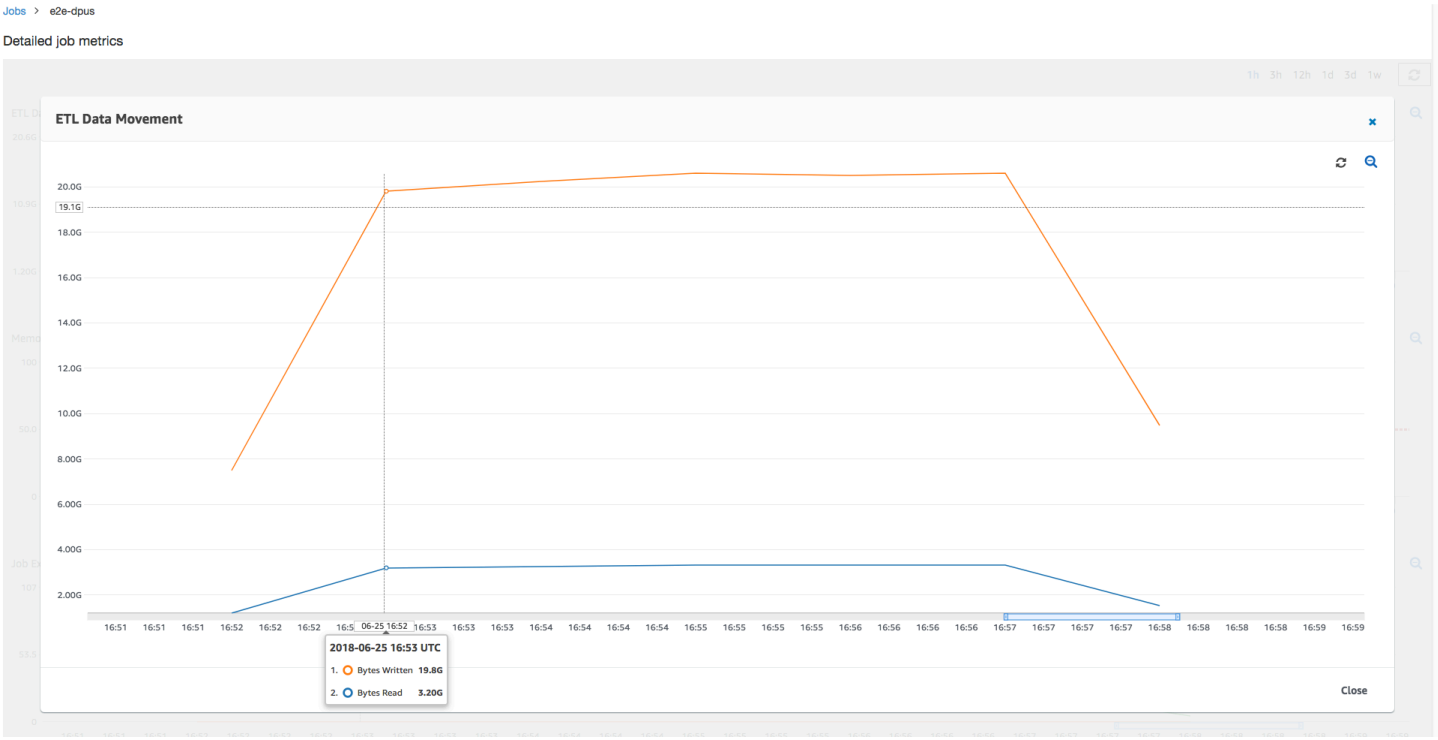
Por contraste, el número de ejecutores que se ejecutan activamente mide cuántos ejecutores ejecutan tareas Apache Spark activas. A medida que progresa el trabajo, el número máximo de ejecutores necesarios puede cambiar y suele reducirse hacia el final del trabajo al disminuir la cola de tareas pendientes.

La línea roja horizontal del siguiente gráfico muestra el número máximo de ejecutores asignados, que depende del número de DPU que asigna para el trabajo. En este caso, asigna 10 DPU para la ejecución de trabajo. Una DPU está reservada para administración. Nueve DPU ejecutan dos ejecutores cada una y un ejecutor se reserva para el controlador Spark. El controlador Spark se ejecuta dentro de la aplicación principal. Así pues, el número máximo de ejecutores asignados es 2 x 9 -1 = 17 ejecutores.



Tal como se muestra en el gráfico, el número máximo de ejecutores necesarios parte de 107 al principio del trabajo, mientras que el número de ejecutores activos sigue siendo 17. Este es igual al número máximo de ejecutores asignados con 10 DPU. La relación entre el número máximo de ejecutores necesarios y el número máximo de ejecutores asignados (que añade 1 a ambos para el controlador Spark) le ofrece el factor de aprovisionamiento deficiente: 108/18 = 6x. Puede aprovisionar 6 (por debajo de la proporción de aprovisionamiento)* 9 (capacidad de DPU actual - 1)+1 DPU = 55 DPU para escalar en horizontal el trabajo a fin de ejecutarlo con el máximo paralelismo y finalizar con mayor rapidez.

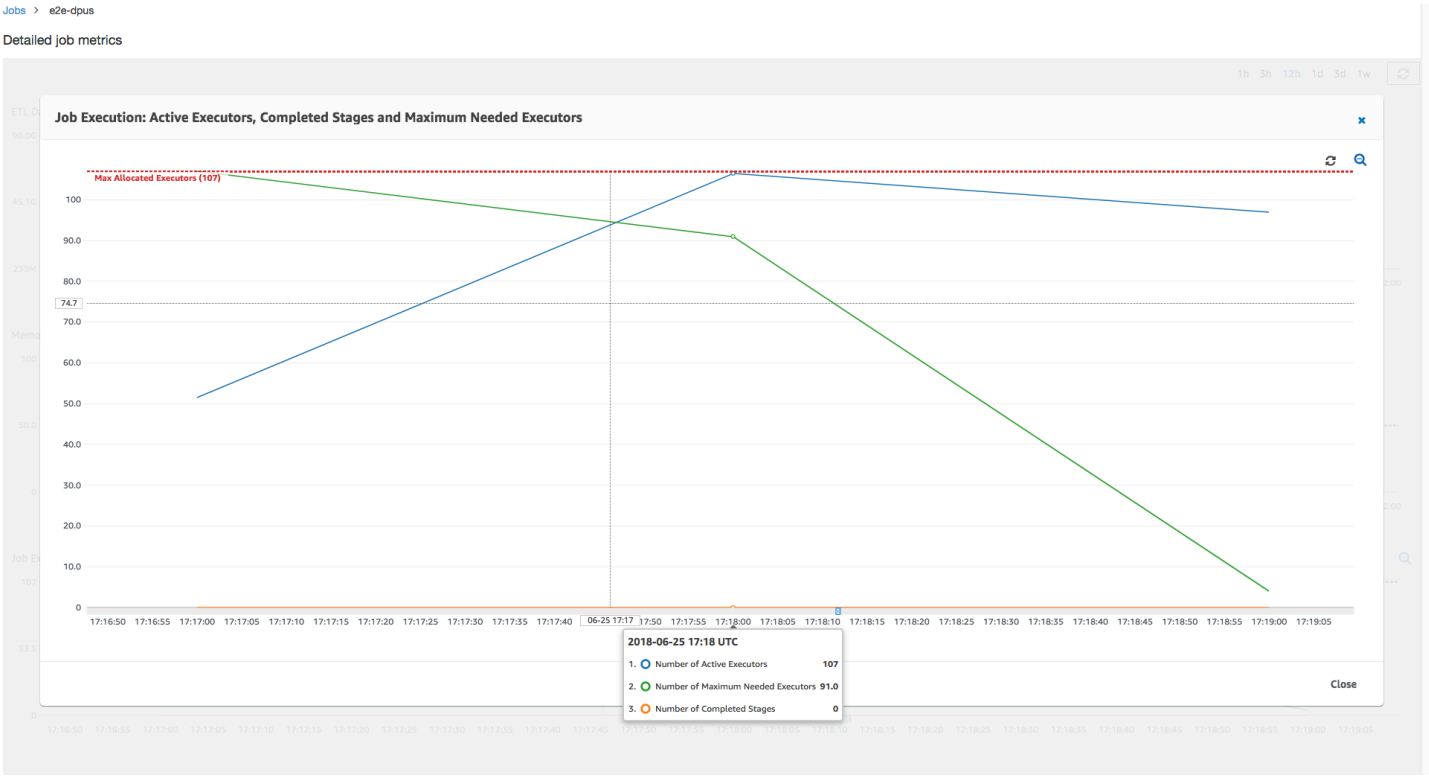
La consola de AWS Glue muestra las métricas detalladas de trabajos como una línea estática que representa el número original de ejecutores máximos asignados. La consola calcula los ejecutores máximos asignados a partir de la definición de trabajo de las métricas. En cambio, para las métricas detalladas de ejecución de trabajos, la consola calcula los ejecutores máximos asignados a partir de la configuración de ejecución de trabajos, específicamente las DPU asignadas a la ejecución de trabajos. Para ver las métricas de una ejecución de trabajo individual, seleccione la ejecución de trabajo y elija View run metrics (Ver métricas de ejecución).



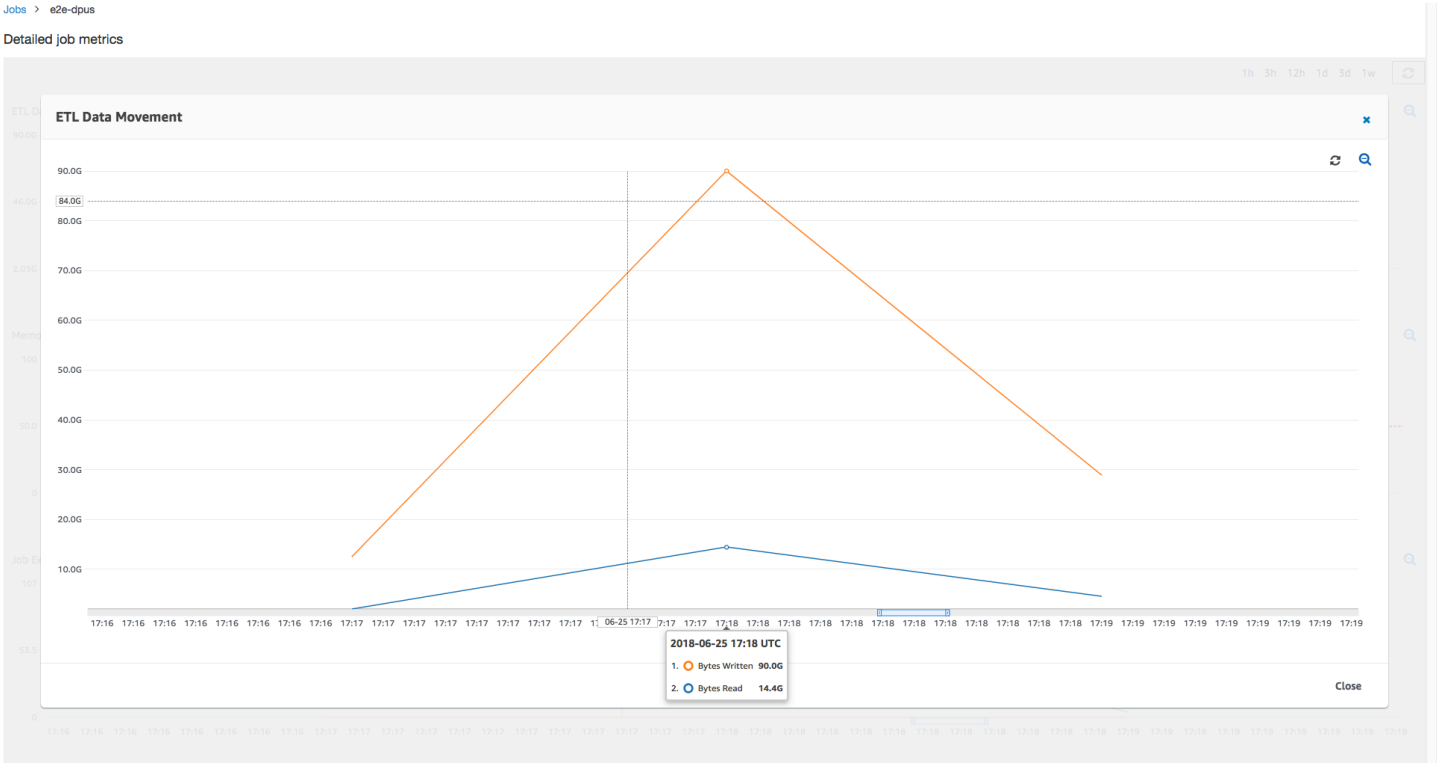
Al examinar los bytes de Amazon S3 leídos y escritos, observe que el trabajo dedica los seis minutos a la streaming de datos desde Amazon S3 y a escribir dichos datos en paralelo. Todos los núcleos de las DPU asignadas leen y escriben en Amazon S3. El número máximo de ejecutores necesarios, 107, también coincide con el número de archivos en la ruta de entrada de Amazon S3, 428. Cada ejecutor puede lanzar cuatro tareas Spark para procesar cuatro archivos de entrada (gzip JSON).

Determinar la capacidad de DPU óptima

En función de los resultados de la ejecución de trabajo anterior, puede aumentar el número total de DPU asignadas a 55, así como consultar el rendimiento del trabajo. El trabajo finaliza en menos de tres minutos, la mitad del tiempo que requería anteriormente. El escalado en horizontal del trabajo no es lineal en este caso porque se trata de un trabajo de corta ejecución. Los trabajos con tareas de larga duración o un gran número de tareas (un gran número correspondiente al máximo de ejecutores necesarios) se benefician de una aceleración de rendimiento de escalado en horizontal de DPU casi lineal.



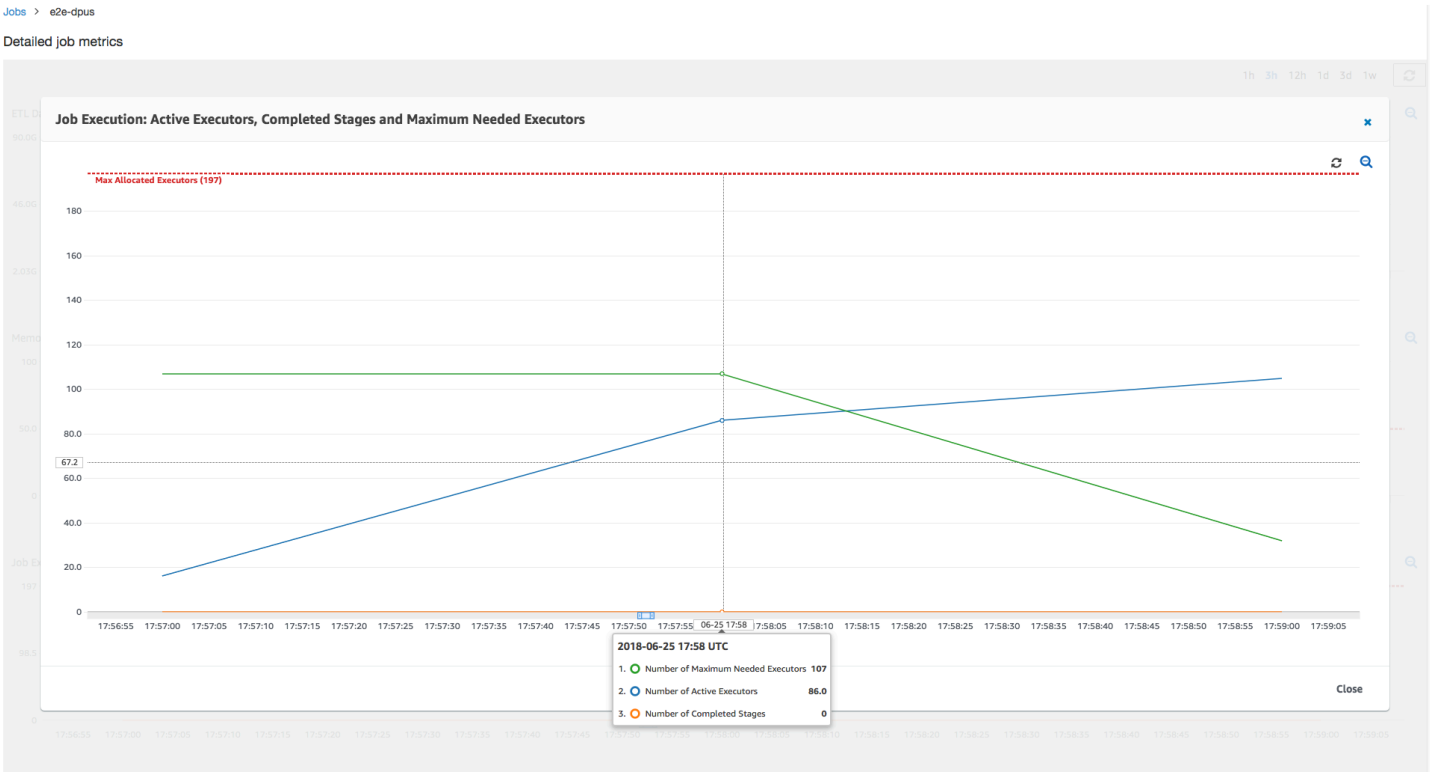
Tal como se muestra en la imagen anterior, el número total de ejecutores activos alcanza el máximo asignado, 107 ejecutores. Del mismo modo, el número máximo de ejecutores necesarios nunca supera el número máximo de ejecutores asignados. El número máximo de ejecutores necesarios se calcula a partir de los recuentos de tareas pendientes y tareas que se ejecutan activamente, por lo que podría ser inferior al número de ejecutores activos. Esto se debe a que puede haber ejecutores que estén inactivos parcial o completamente durante un corto periodo de tiempo y no se hayan retirado aún.



Esta ejecución de trabajo usa seis veces más ejecutores para leer y escribir desde Amazon S3 en paralelo. Como resultado, esta ejecución de trabajo usa más ancho de banda de Amazon S3 tanto para las lecturas como para las escrituras, y finaliza con mayor rapidez.

Identificar DPU sobreaprovisionadas

A continuación, puede determinar si escalar en horizontal el trabajo con 100 DPU (99 x 2 = 198 ejecutores) ayuda a escalar más en horizontal. Tal como se muestra en el siguiente gráfico, el trabajo sigue tardando tres minutos en finalizar. Del mismo modo, el trabajo no escala en horizontal más allá de 107 ejecutores (configuración de 55 DPU) y los 91 ejecutores restantes están sobreaprovisionados y no se han utilizado en absoluto. Esto muestra que es posible que aumentar el número de DPU no siempre mejore el rendimiento, como se desprende del número máximo de ejecutores necesarios.



Comparar las diferencias de tiempo

Las tres ejecuciones de trabajo mostradas en la siguiente tabla resumen los tiempos de ejecución de trabajo para 10 DPU, 55 DPU y 100 DPU. Puede encontrar la capacidad de DPU para mejorar el tiempo de ejecución de trabajo mediante las estimaciones que estableció monitorizando la primera ejecución de trabajo.

ID del trabajo	Número de DPU	Hora de ejecución
jr_c894524c8ef5048a4d9...	10	6 min.
jr_1a466cf2575e7ffe6856...	55	3 min.
jr_34fa1ed4c6aa9ff0a814...	100	3 min.

Trabajos ETL de streaming en AWS Glue

Puede crear trabajos de extracción, transformación y carga (ETL) de streaming que se ejecuten continuamente, consuman datos de orígenes de streaming como Amazon Kinesis Data Streams, Apache Kafka y Amazon Managed Streaming for Apache Kafka (Amazon MSK). Los trabajos limpian

y transforman los datos y, a continuación, cargan los resultados en los lagos de datos de Amazon S3 o en los almacenes de datos JDBC.

Además, puede generar datos para los flujos de Amazon Kinesis Data Streams. Esta función solo está disponible al escribir AWS Glue guiones. Para obtener más información, consulte the section called “Conexión de Kinesis”.

De forma predeterminada, AWS Glue procesa y escribe datos en periodos de 100 segundos. Esto permite que los datos se procesen de forma eficiente y permite que las agregaciones se realicen en los datos que lleguen más tarde de lo previsto. Puede modificar este tamaño de ventana para aumentar la puntualidad o la precisión de la agregación. Los trabajos de streaming de AWS Glue utilizan puntos de control en lugar de marcadores de trabajo para rastrear los datos leídos.

> **Note**
>
> AWS Glue factura por hora para trabajos de ETL de streaming mientras se están ejecutando.

En este vídeo se analizan los retos de coste de la transmisión por ETL y las funciones que permiten ahorrar costes. AWS Glue

La creación de un trabajo ETL de streaming consta de los siguientes pasos:

1. Para un origen de streaming de Apache Kafka, cree una conexión de AWS Glue al origen de Kafka o al clúster de Amazon MSK.
2. Cree manualmente una tabla del Catálogo de datos para el origen de streaming.
3. Cree un trabajo ETL para el origen de datos de streaming. Defina propiedades de trabajo específicas de streaming y proporcione su propio script o también puede modificar el script generado.

Para obtener más información, consulte ETL de streaming en AWS Glue.

Al crear un trabajo de ETL de streaming para Amazon Kinesis Data Streams, no es necesario crear una conexión de AWS Glue. Sin embargo, si hay una conexión asociada al trabajo de ETL de streaming de AWS Glue que tenga Kinesis Data Streams como origen, se requiere un punto de enlace de la nube privada virtual (VPC) para Kinesis. Para obtener más información, consulte Creación de un punto de conexión de interfaz en la Guía del usuario de Amazon VPC. Al especificar un flujo de Amazon Kinesis Data Streams en otra cuenta, debe configurar los roles y las políticas

para permitir el acceso entre cuentas. Para obtener más información, consulte Ejemplo: leer desde un flujo de Kinesis en una cuenta diferente.

Los trabajos de ETL de streaming en AWS Glue pueden detectar automáticamente datos comprimidos, descomprimir de forma transparente los datos de streaming, realizar las transformaciones habituales en el origen de entrada y cargarlos en el almacén de salida.

AWS Glue admite la descompresión automática para los siguientes tipos de compresión dado el formato de entrada:

Tipo de compresión	Archivo Avro	Datos de Avro	JSON	CSV	Grok
BZIP2	Sí	Sí	Sí	Sí	Sí
GZIP	No	Sí	Sí	Sí	Sí
SNAPPY	Sí (Snappy sin formato)	Sí (con encuadre de Snappy)	Sí (con encuadre de Snappy)	Sí (con encuadre de Snappy)	Sí (con encuadre de Snappy)
XZ	Sí	Sí	Sí	Sí	Sí
ZSTD	Sí	No	No	No	No
DEFLATE	Sí	Sí	Sí	Sí	Sí

Temas

- Creación de una conexión de AWS Glue para un flujo de datos Apache Kafka
- Creación de una tabla del Catálogo de datos para un origen de streaming
- Notas y restricciones para orígenes de streaming de Avro
- Aplicación de patrones Grok a orígenes de streaming
- Definición de propiedades de trabajo para un trabajo ETL de streaming
- Notas y restricciones de ETL de streaming

Creación de una conexión de AWS Glue para un flujo de datos Apache Kafka

Para leer en una transmisión de Apache Kafka, debe crear una conexión de AWS Glue.

Para crear una conexión de AWS Glue para un origen de Kafka (consola)

1. Abra la AWS Glue consola en https://console.aws.amazon.com/glue/.
2. En el panel de navegación, en Data catalog (Catálogo de datos), elija Connections (Conexiones).
3. Elija Agregar conexión y, en la página Configurar las propiedades de la conexión escriba un nombre de conexión.

   > **Note**
   >
   > Para obtener más información acerca de cómo especificar propiedades de conexión, consulte Propiedades de conexión de AWS Glue.

4. En Tipo de conexión, elija Kafka.
5. Para Kafka bootstrap servers URLs (URL de servidores de arranque Kafka), ingrese el host y el número de puerto de los agentes de arranque para su clúster de Amazon MSK o Apache Kafka. Utilice sólo los puntos de enlace de Transport Layer Security (TLS) para establecer la conexión inicial con el clúster de Kafka. No se admiten puntos de enlace de texto no cifrado.

   A continuación se ofrece una lista de pares de nombre de host y número de puerto para un clúster de Amazon MSK a modo de ejemplo.

   ```
   myserver1.kafka.us-east-1.amazonaws.com:9094,myserver2.kafka.us-
   east-1.amazonaws.com:9094,
   myserver3.kafka.us-east-1.amazonaws.com:9094
   ```

   Para obtener más información sobre cómo obtener la información del agente de arranque, consulte Obtener los agentes de arranque para un clúster de Amazon MSK en la Guía para desarrolladores de Amazon Managed Streaming for Apache Kafka.
6. Si desea una conexión segura al origen de datos de Kafka, seleccione Require SSL connection (Solicitar conexión SSL), y para Kafka private CA certificate location (Ubicación del certificado de CA privado de Kafka), ingrese una ruta válida de Amazon S3 a un certificado SSL personalizado.

Para una conexión SSL a Kafka autoadministrado, el certificado personalizado es obligatorio. Es opcional para Amazon MSK.

Para obtener más información sobre cómo especificar un certificado personalizado para Kafka, consulte the section called “Propiedades de las conexiones SSL”.

7. Utilice AWS Glue Studio o la AWS CLI para especificar un método de autenticación de cliente de Kafka. Para acceder, AWS Glue Studio seleccione una opción AWS Gluedel menú ETL en el panel de navegación izquierdo.

   Para obtener más información acerca de los métodos de autenticación de cliente Kafka, consulte Propiedades de conexión de AWS Glue Kafka para autenticación de clientes .
8. Si lo desea, escriba una descripción y, a continuación, elija Next (Siguiente).
9. Para un clúster de Amazon MSK, especifique la nube virtual privada (VPC), su subred y el grupo de seguridad. La información de la VPC es opcional para Kafka autoadministrado.
10. Elija Next (Siguiente) para revisar todas las propiedades de conexión y, a continuación, elija Finish (Finalizar).

Para obtener más información acerca de las conexiones de AWS Glue, consulte Conexión a datos.

Propiedades de conexión de AWS Glue Kafka para autenticación de clientes

Autenticación SASL/GSSAPI (Kerberos)

La elección de este método de autenticación le permitirá especificar propiedades de Kerberos.

Keytab de Kerberos

Seleccione la ubicación del archivo keytab. Una keytab almacena claves a largo plazo para uno o varias entidades principales. Para obtener más información, consulte MIT Kerberos Documentation: Keytab (Documentación de MIT Kerberos: Keytab).

Archivo Kerberos krb5.conf

Seleccione el archivo krb5.conf. Contiene el dominio predeterminado (una red lógica, similar a un dominio, que define un grupo de sistemas bajo el mismo KDC) y la ubicación del servidor KDC. Para obtener más información, consulte MIT Kerberos Documentation: krb5.conf (Documentación de MIT Kerberos: krb5.conf).

Nombre principal y nombre de servicio de Kerberos

Ingrese el nombre principal y el de servicio de Kerberos. Para obtener más información, consulte la documentación de Kerberos de MIT: principal de Kerberos.

Autenticación SASL/SCRAM-SHA-512

La elección de este método de autenticación le permitirá especificar credenciales de autenticación.

AWS Secrets Manager

Busque el token en el cuadro de búsqueda mediante el nombre o ARN.

Nombre de usuario y contraseña del proveedor directamente

Busque el token en el cuadro de búsqueda mediante el nombre o ARN.

Autenticación SSL del cliente

La elección de este método de autenticación le permite seleccionar la ubicación del almacén de claves del cliente Kafka navegando por Amazon S3. Opcionalmente, puede ingresar la contraseña del almacén de claves del cliente Kafka y la contraseña de clave de cliente Kafka.

Autenticación de IAM

Este método de autenticación no requiere especificaciones adicionales y solo se aplica cuando la fuente de streaming es Kafka para MSK.

Autenticación de SASL/PLAIN

Al elegir este método de autenticación, puede especificar las credenciales de autenticación.

Creación de una tabla del Catálogo de datos para un origen de streaming

Es posible crear de forma manual una tabla del Catálogo de datos que especifique las propiedades del flujo de datos de origen, incluido el esquema de datos, para un origen de streaming. Esta tabla se utiliza como origen de datos para el trabajo ETL de streaming.

Si no conoce el esquema de los datos en el flujo de datos de origen, puede crear la tabla sin un esquema. A continuación, cuando cree el trabajo de ETL de streaming, puede habilitar la función detección de esquema de AWS Glue. AWS Glue determinará el esquema a partir de los datos de streaming.

Utilice la AWS Glueconsola, el AWS Command Line Interface (AWS CLI) o la AWS Glue API para crear la tabla. Para obtener información sobre cómo crear una tabla manualmente con la consola de AWS Glue, consulte the section called “Creación de tablas”.

> **Note**
>
> No puedes usar la AWS Lake Formation consola para crear la tabla; debes usarla. AWS Glue

Tenga en cuenta también la siguiente información para los orígenes de streaming en formato Avro o para los datos de registro a los que puede aplicar patrones Grok.

- the section called “Notas y restricciones para orígenes de streaming de Avro”
- the section called “Aplicación de patrones Grok a orígenes de streaming”

Temas

- Origen de datos de Kinesis
- Origen de datos de Kafka
- Origen de tablas de AWS Glue Schema Registry

Origen de datos de Kinesis

Al crear la tabla, configure las siguientes propiedades de ETL de streaming (consola).

Tipo de origen

Kinesis

Para un origen de Kinesis en la misma cuenta:

Región

La AWS región en la que reside el servicio Amazon Kinesis Data Streams. La región y el nombre del flujo de Kinesis se traducen juntos a un ARN de flujo.

Ejemplo: https://kinesis.us-east-1.amazonaws.com

Nombre de flujo de Kinesis

Nombre de la transmisión, tal como se describe en Creación de una transmisión en la Guía para desarrolladores de Amazon Kinesis Data Streams.

Para un origen de Kinesis en otra cuenta, consulte este ejemplo para configurar los roles y políticas que permitan el acceso entre cuentas. Configure estas opciones:

ARN de flujo

El ARN del flujo de datos de Kinesis con el que está registrado el consumidor. Para obtener más información, consulte Nombres de recursos (ARN) y espacios de nombres de AWS servicios de Amazon en. Referencia general de AWS

ARN de rol asumido

El nombre de recurso de Amazon (ARN) del rol que se asignará.

Nombre de la sesión (opcional)

Un identificador para la sesión del rol asumido.

Utilice el nombre de la sesión del rol para identificar de forma única una sesión cuando el mismo rol es asumido por diferentes entidades o por diferentes razones. En escenarios entre cuentas, el nombre de la sesión del rol es visible para la cuenta que es titular del rol. Esta cuenta podrá registrar el rol. El nombre de la sesión del rol también se utiliza en el ARN de la entidad principal del rol asumido. Esto significa que las solicitudes de API multicuenta posteriores que utilicen las credenciales de seguridad temporales expondrán el nombre de la sesión del rol a la cuenta externa en sus registros. AWS CloudTrail

Para configurar las propiedades de ETL de streaming para Amazon Kinesis Data Streams (API de AWS Glue o AWS CLI)

- Para configurar las propiedades de ETL de streaming para un origen de Kinesis en la misma cuenta, especifique los parámetros `streamName` y `endpointUrl` en la estructura `StorageDescriptor` de la operación de la API `CreateTable` o el comando de la CLI `create_table`.

```
"StorageDescriptor": {
 "Parameters": {
  "typeOfData": "kinesis",
  "streamName": "sample-stream",
  "endpointUrl": "https://kinesis.us-east-1.amazonaws.com"
 }
 ...
}
```

O bien, especifique `streamARN`.

Example

```
"StorageDescriptor": {
 "Parameters": {
  "typeOfData": "kinesis",
  "streamARN": "arn:aws:kinesis:us-east-1:123456789:stream/sample-stream"
 }
 ...
}
```

- Para configurar las propiedades de ETL de streaming para un origen de Kinesis en otra cuenta, especifique los parámetros `streamARN`, `awsSTSRoleARN` y `awsSTSSessionName` (opcional) en la estructura `StorageDescriptor` en la operación de la API `CreateTable` o el comando de la CLI `create_table`.

```
"StorageDescriptor": {
 "Parameters": {
  "typeOfData": "kinesis",
  "streamARN": "arn:aws:kinesis:us-east-1:123456789:stream/sample-stream",
  "awsSTSRoleARN": "arn:aws:iam::123456789:role/sample-assume-role-arn",
  "awsSTSSessionName": "optional-session"
 }
 ...
}
```

Origen de datos de Kafka

Al crear la tabla, configure las siguientes propiedades de ETL de streaming (consola).

Tipo de origen

Kafka

Para un origen de Kafka:

Nombre del tema

Nombre del tema como se especifica en Kafka.

Connection

Una conexión de AWS Glue que hace referencia a un origen de Kafka, como se describe en the section called “Crear una conexión para un flujo de datos Kafka”.

Origen de tablas de AWS Glue Schema Registry

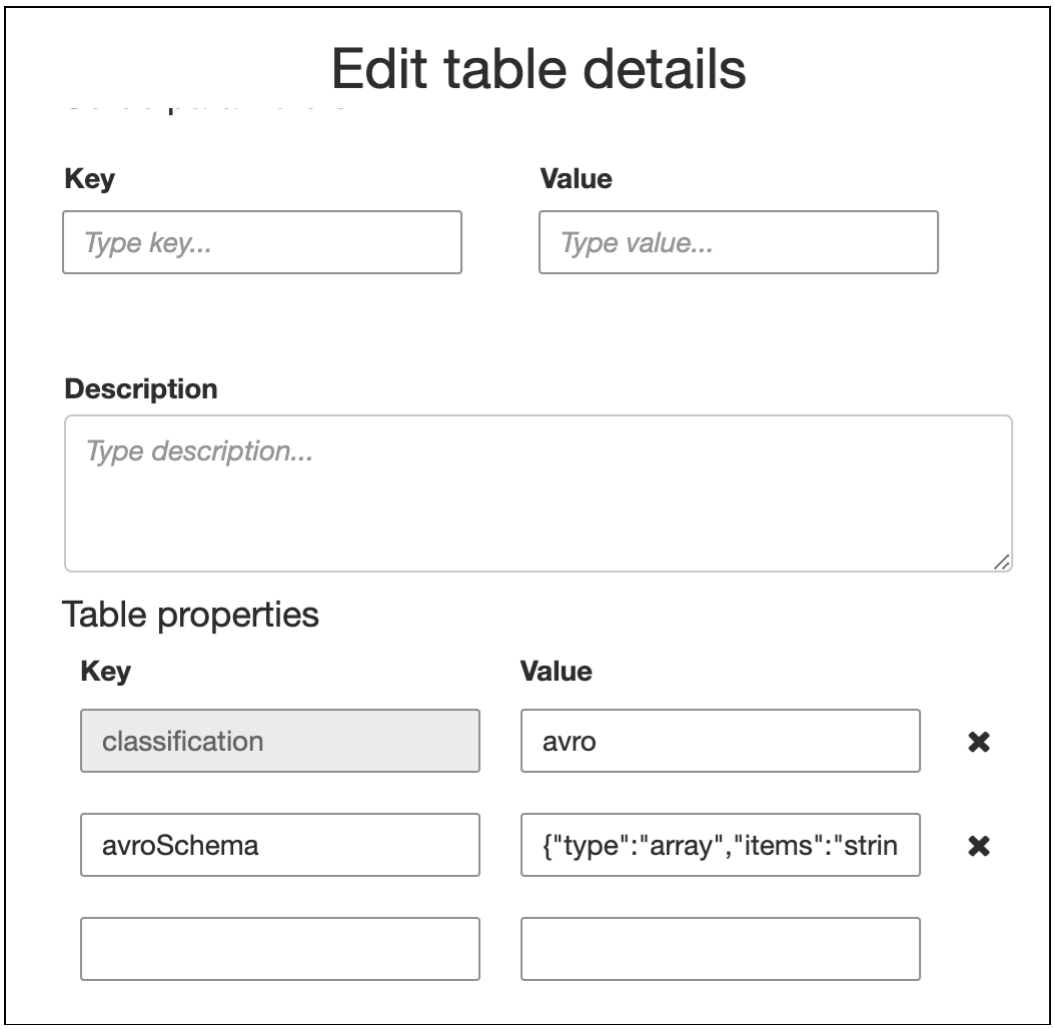
A fin de utilizar AWS Glue Schema Registry para trabajos de streaming, siga las instrucciones de Caso de uso: AWS Glue Data Catalog para crear o actualizar una tabla de Schema Registry.

Actualmente, el streaming de AWS Glue solo soporta el formato Avro de Glue Schema Registry con inferencia de esquema establecida en `false`.

Notas y restricciones para orígenes de streaming de Avro

Las siguientes notas y restricciones corresponden a los orígenes de streaming en formato Avro:

- Cuando se activa la detección de esquemas, el esquema de Avro debe incluirse en la carga. Cuando se desactiva, la carga debe contener solo datos.
- Algunos tipos de datos de Avro no se admiten en marcos dinámicos. No se pueden especificar estos tipos de datos al definir el esquema con la página Define a schema (Definir un esquema) en el asistente de creación de tabla en la consola de AWS Glue. Durante la detección de esquemas, los tipos no soportados en el esquema Avro se convierten en tipos soportados de la siguiente manera:
  - `EnumType => StringType`
  - `FixedType => BinaryType`
  - `UnionType => StructType`
- Si define el esquema de tabla mediante la página Define a schema (Definir un esquema) en la consola, el tipo de elemento raíz implícito para el esquema es `record`. Si desea un tipo de elemento raíz distinto de `record`, por ejemplo, `array` or `map`, no puede especificar el esquema mediante la página Define a schema (Definir un esquema). En su lugar, debe omitir esa página y especificar el esquema como propiedad de tabla o dentro del script de ETL.
  - Para especificar el esquema en las propiedades de la tabla, complete el asistente de creación de tabla, edite los detalles de la tabla y agregue un nuevo par clave-valor en Table properties (Propiedades de la tabla). Utilice la clave `avroSchema`, e ingrese un objeto JSON de esquema para el valor, tal y como se muestra en la siguiente captura de pantalla.



- Para especificar el esquema en el script de ETL, modifique la instrucción de asignación `datasource0` y agregue la clave `avroSchema` a la instrucción `additional_options`, tal y como se muestra en los siguientes ejemplos de Python y Scala.

  Python

  ```
  SCHEMA_STRING = '{"type":"array","items":"string"}'
  datasource0 = glueContext.create_data_frame.from_catalog(database =
   "database", table_name = "table_name", transformation_ctx = "datasource0",
   additional_options = {"startingPosition": "TRIM_HORIZON", "inferSchema":
   "false", "avroSchema": SCHEMA_STRING})
  ```

  Scala

  ```
  val SCHEMA_STRING = """{"type":"array","items":"string"}"""
  val datasource0 = glueContext.getCatalogSource(database = "database", tableName
   = "table_name", redshiftTmpDir = "", transformationContext = "datasource0",
  ```

```
additionalOptions = JsonOptions(s"""{"startingPosition": "TRIM_HORIZON",
"inferSchema": "false", "avroSchema":"$SCHEMA_STRING"}""")).getDataFrame()
```

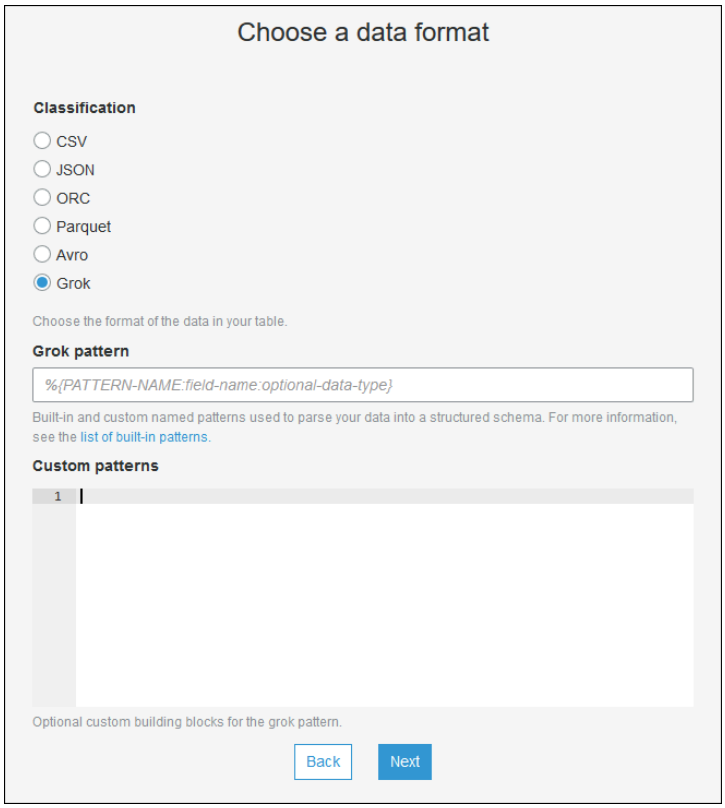
Aplicación de patrones Grok a orígenes de streaming

Puede crear un trabajo de ETL de streaming para un origen de datos de registro y utilizar patrones Grok para convertir los registros en datos estructurados. Luego, el trabajo de ETL procesa los datos como un origen de datos estructurados. Especifique los patrones Grok que se aplicarán al crear la tabla del Catálogo de datos para el origen de streaming.

Para obtener información acerca de los patrones Grok y los valores de la cadena de patrones personalizados, consulte Escritura de clasificadores personalizados de Grok.

Para agregar patrones Grok a la tabla del Catálogo de datos (consola)

- Utilice el asistente de creación de tablas y cree la tabla con los parámetros especificados en the section called “Creación de una tabla del Catálogo de datos para un origen de streaming”. Especifique el formato de datos como Grok, complete el campo Grok pattern (Patrón de Grok) y, opcionalmente, agregue patrones personalizados en Custom patterns (Patrones personalizados) (opcional).



Presione Enter (Intro) después de cada patrón personalizado.

Para agregar patrones grok a la tabla del Catálogo de datos (API de AWS Glue o AWS CLI)

- Agregue el parámetro `GrokPattern` y, opcionalmente, el parámetro `CustomPatterns` a la operación de API `CreateTable` o al comando de la CLI `create_table`.

```
 "Parameters": {
...
    "grokPattern": "string",
    "grokCustomPatterns": "string",
...
},
```

Indique `grokCustomPatterns` como una cadena y use “\n” como separador entre patrones.

El siguiente ejemplo muestra cómo especificar estos parámetros.

Example

```
"parameters": {
...
    "grokPattern": "%{USERNAME:username} %{DIGIT:digit:int}",
    "grokCustomPatterns": "digit \d",
...
}
```

Definición de propiedades de trabajo para un trabajo ETL de streaming

Cuando defina un trabajo de ETL de streaming en la consola de AWS Glue, proporcione las siguientes propiedades específicas de los flujos. Para obtener propiedades de trabajo adicionales, consulte Definición de propiedades de trabajo para trabajos de Spark.

Rol de IAM

Especifique la función AWS Identity and Access Management (IAM) que se utiliza para autorizar los recursos que se utilizan para ejecutar el trabajo, acceder a las fuentes de streaming y acceder a los almacenes de datos de destino.

Para acceder a Amazon Kinesis Data Streams, asocie `AmazonKinesisFullAccess` AWS la política gestionada al rol o adjunte una política de IAM similar que permita un acceso más detallado. Para obtener información sobre políticas de ejemplo, consulte Control del acceso a los recursos de Amazon Kinesis Data Streams mediante IAM.

Para obtener más información acerca de los permisos para ejecutar trabajos en AWS Glue, consulte Gestión de identidad y acceso para AWS Glue.

Tipo

Elija Spark Streaming.

Versión de AWS Glue

La versión de AWS Glue determina las versiones de Apache Spark y Python o Scala que están disponibles para el trabajo. Elija una selección que especifique la versión de Python o Scala disponible para el trabajo. AWS Glue La versión 2.0 con soporte de Python 3 es la predeterminada para los trabajos de ETL de streaming.

Periodo de mantenimiento

Especifica una ventana en la que se puede reiniciar un trabajo de streaming. Consulte the section called “Periodos de mantenimiento”.

Job timeout (Tiempo de espera de flujo de trabajo)

Si lo desea, escriba una duración en minutos. El valor predeterminado está en blanco.

- Los trabajos de streaming deben tener un valor de tiempo de espera inferior a 7 días o 10080 minutos.
- Si el valor se deja en blanco, el trabajo se reiniciará después de 7 días, si no ha establecido un período de mantenimiento. Si ha configurado un período de mantenimiento, el trabajo se reiniciará durante el período de mantenimiento transcurridos 7 días.

Origen de datos

Elimine la tabla que creó en the section called “Creación de una tabla del Catálogo de datos para un origen de streaming”.

Destino de datos

Realice una de las siguientes acciones siguientes:

- Elija Crear tablas en el destino de datos y especifique las siguientes propiedades de destino de datos.

Almacén de datos

Seleccione Amazon S3 o JDBC.

Formato

Elija cualquier formato. Todos son compatibles para streaming.

- Elija Use tables in the data catalog and update your data target (Utilizar tablas en el Catálogo de datos y actualizar el destino de los datos) y elija una tabla para un almacén de datos JDBC.

Definición de esquema de salida

Realice una de las siguientes acciones siguientes:

- Seleccione Automatically detect schema of each record (Detectar automáticamente el esquema de cada registro) para habilitar la detección de esquemas. AWS Glue determina el esquema a partir de los datos de streaming.
- Seleccione Specify output schema for all records (Especificar esquema de salida para todos los registros) para utilizar la transformación de la función de aplicar mapeo a fin de definir el esquema de salida.

Script

Si lo desea, proporcione su propio script o modifique el script generado para realizar operaciones compatibles con el motor Apache Spark Structured Streaming. Para obtener información sobre las operaciones disponibles, consulte Operaciones en streaming DataFrames /Datasets.

Notas y restricciones de ETL de streaming

Tenga en cuenta las siguientes notas y restricciones:

- La descompresión automática para los trabajos de ETL de streaming en AWS Glue solo está disponible para los tipos de compresión admitidos. Tenga también en cuenta lo siguiente:
  - Con encuadre de Snappy se refiere al formato de encuadre oficial para Snappy.
  - Deflate es compatible con la versión 3.0 de Glue, no con la versión 2.0.
- Cuando se utiliza la detección de esquemas, no se pueden realizar combinaciones de datos de streaming.
- Los trabajos ETL de streaming de AWS Glue no admiten el tipo de datos de la Unión para el Registro de esquemas de AWS Glue con formato Avro.

- Su script de ETL puede usar las transformaciones incorporadas de AWS Glue y las transformaciones nativas de Apache Spark Structured Streaming. Para obtener más información, consulte Operaciones de transmisión de DataFrames /Datasets en el sitio web de Apache Spark o. AWS Glue PySpark transforma la referencia
- Los trabajos de ETL de streaming de AWS Glue utilizan puntos de comprobación para realizar un seguimiento de los datos que se han leído. Por lo tanto, un trabajo detenido y reiniciado comienza donde lo dejó en la transmisión. Si desea volver a procesar los datos, puede eliminar la carpeta de puntos de control a la que se hace referencia en el script.
- No se admiten los marcadores de trabajo.
- Para la función de distribución mejorada de Kinesis Data Streams en un trabajo, consulte the section called “Uso de una distribución mejorada en los trabajos de streaming de Kinesis”.
- Si utiliza una tabla del Catálogo de datos creada a partir de AWS Glue Schema Registry, cuando se encuentre disponible una versión de esquema nueva, para reflejar el esquema nuevo, debe realizar lo siguiente:
  1. Detenga los trabajos asociados a la tabla.
  2. Actualice el esquema de la tabla del Catálogo de datos.
  3. Reinicie los trabajos asociados a la tabla.

Coincidencia de registros con FindMatches de AWS Lake Formation

> **Note**
>
> La coincidencia de registros no está disponible por el momento en las siguientes regiones de la consola de AWS Glue: Oriente Medio (EAU), Europa (España) (`eu-south-2`) y Europa (Zúrich) (`eu-central-2`).

AWS Lake Formation proporciona las capacidades de machine learning para crear transformaciones personalizadas para borrar sus datos. En la actualidad, hay una transformación disponible denominada FindMatches. La transformación FindMatches le permite identificar registros duplicados o coincidentes en el conjunto de datos, incluso cuando los registros no tienen un identificador único común y no coinciden exactamente los campos. Esto no requiere escribir ningún código ni saber cómo funciona machine learning. FindMatches puede resultar útil en muchos problemas diferentes, como:

- Coincidencia de clientes: vinculación de registros de clientes en diferentes bases de datos de clientes, incluso cuando muchos campos de clientes no coincidan exactamente en las bases de datos (por ejemplo, ortografía de nombres diferentes, diferencias de direcciones, datos que faltan o no son precisos, etc.).
- Coincidencia de productos: cotejar los productos del catálogo con otros orígenes de productos, como el catálogo de productos con el catálogo de un competidor, donde las entradas se estructuran de forma diferente.
- Mejora de la detección de fraudes: identificar cuentas de cliente duplicadas, determinar cuándo una cuenta recién creada es (o podría ser) una coincidencia para un usuario fraudulento conocido anteriormente.
- Otros problemas de coincidencia: coincidencia de direcciones, películas, listas de partes, etc. En general, si un ser humano pudiera examinar las filas de la base de datos y determinar que eran una coincidencia, existe una muy buena posibilidad de que la transformación FindMatches pueda ayudarle.

Puede crear estas transformaciones cuando cree un trabajo. La transformación que se crea se basa en un esquema de almacén de datos de origen y datos de ejemplo del conjunto de datos de origen que se etiquetan (a este proceso se lo denomina “enseñar” a una transformación). Los registros que se etiquetan deben estar presentes en el conjunto de datos de origen. En este proceso, se genera un archivo que se etiqueta y que luego se vuelve a cargar, y del que la transformación aprende en cierto modo. Después de enseñar a su transformación, puede invocarla desde su trabajo de AWS Glue basado en Spark (PySpark o Scala Spark) y utilizarla en otros scripts con un almacén de datos de origen compatible.

Una vez que se crea la transformación, se almacena en AWS Glue. En la consola de AWS Glue, puede administrar las transformaciones que cree. En el panel de navegación, en Integración de datos y ETL, en Herramientas de clasificación de datos > Coincidencia de registros, puede editar y entrenar su transformación de machine learning. Para obtener más información sobre la administración de transformaciones en la consola, consulte Uso de transformaciones de machine learning en la consola de AWS Glue.

> **Note**
>
> Los trabajos de FindMatches de AWS Glue versión 2.0 utilizan el bucket de Amazon S3 `aws-glue-temp-<accountID>-<region>` para almacenar archivos temporales mientras la transformación procesa datos. Puede eliminar estos datos una vez finalizada la ejecución,

ya sea en forma manual o mediante la configuración de una regla de ciclo de vida de Amazon S3.

Tipos de transformaciones de machine learning

Puede crear transformaciones de machine learning para limpiar sus datos. Puede llamar a estas transformaciones desde el script de ETL. Los datos pasan de transformación en transformación en una estructura de datos denominada DynamicFrame, que es una extensión de `DataFrame` de Apache Spark SQL. `DynamicFrame` contiene sus datos y usted hace referencia a su esquema para procesar los datos.

Los tipos de transformaciones de machine learning disponibles son los siguientes:

Búsqueda de coincidencias

Permite buscar registros duplicados en los datos de origen. Enseñe esta transformación de machine learning mediante el etiquetado de conjuntos de datos de ejemplo para indicar las filas que coinciden. La transformación de machine learning aprende qué filas deben ser coincidencias a medida que usted se lo va enseñando utilizando datos etiquetados de ejemplo. En función de cómo configure la transformación, se genera una de las siguientes salidas:

- Una copia de la tabla de entrada además de una columna `match_id` completada con valores que indican conjuntos de coincidencia de registros. La columna `match_id` es un identificador arbitrario. Cualquier registro que tenga el mismo `match_id` se ha identificado como coincidente entre sí. Los registros con diferentes `match_id`no coinciden.
- Una copia de la tabla de entrada con las filas duplicadas eliminadas. Si se detectan varios duplicados, se guarda el registro con el menor clave principal.

Búsqueda de coincidencias progresivas

La transformación Buscar coincidencias también se puede configurar para buscar coincidencias en los fotogramas existentes y progresivos y devolver como salida una columna que contiene un ID único por grupo de coincidencias.

Para obtener más información, consulte: Búsqueda de coincidencias progresivas

Uso de la transformación FindMatches

Puede utilizar la transformación `FindMatches` para buscar registros duplicados en los datos de origen. Se genera un archivo de etiquetado o se proporciona para ayudar a enseñar a la transformación.

> **Note**
>
> En la actualidad, las transformaciones de `FindMatches` que utilizan una clave de cifrado personalizada no se soportan en las siguientes regiones:
>
> - Asia-Pacífico (Osaka): `ap-northeast-3`

Para empezar con la transformación FindMatches, puede seguir los pasos que se indican a continuación. Para obtener un ejemplo más avanzado y detallado, consulte el blog sobre macrodatos de AWS: Armonizar los datos con la ML de FindMatches AWS Glue y AWS Lake Formation para crear una vista 360 del cliente.

Introducción al uso de la transformación de FindMatches

Siga estos pasos para iniciar la transformación de `FindMatches`:

1. Cree una tabla en AWS Glue Data Catalog para los datos de origen que tienen que borrarse. Para obtener información sobre cómo crear un rastreador, consulte Trabajar de rastreadores en la consola de AWS Glue.

   Si los datos de origen son un archivo basado en texto como un archivo de valores separados por comas (CSV), tenga en cuenta lo siguiente:

   - Mantenga el archivo de etiquetado y el archivo CSV de registro de entrada en carpetas independientes. De lo contrario, el rastreador de AWS Glue podría considerarlos como varias partes de la misma tabla y crear tablas en el Catálogo de datos de forma incorrecta.
   - A menos que su archivo CSV incluya únicamente caracteres ASCII, asegúrese de que se utiliza UTF-8 sin codificación BOM (byte order mark, marca de orden de bytes) para los archivos CSV. Microsoft Excel a menudo añade una BOM al principio de archivos CSV UTF-8. Para eliminarlo, abra el archivo CSV en un editor de texto y vuelva a guardar el archivo como UTF-8 sin BOM.
2. En la consola AWS Glue, cree un trabajo y elija el tipo de transformación de búsqueda de coincidencias.

> **Important**
>
> La tabla de origen de datos que elija para el trabajo no puede tener más de 100 columnas.

3. AWS Glue Indique a que genere un archivo de etiquetado eligiendo Generate labeling file (Generar archivo de etiquetado). AWS Glue realiza la primera pasada al agrupar registros similares para cada uno de ellos para que `labeling_set_id` pueda revisar esas agrupaciones. Las coincidencias se etiquetan en la columna `label`.
   - Si ya tiene un archivo de etiquetado, es decir, un ejemplo de registros que indican filas de coincidencia, cargue el archivo en Amazon Simple Storage Service (Amazon S3). Para obtener más información sobre el formato del archivo de etiquetado, consulte Formato de archivo de etiquetado. Continúe con el paso 4.
4. Descargue el archivo de etiquetado y etiquete el archivo tal y como se describe en la sección Etiquetado.
5. Cargue el archivo de etiquetado corregido. AWS Glue ejecuta tareas para enseñar a la transformación cómo encontrar coincidencias.

   En la página de lista Machine learning transforms (Transformaciones de machine learning), elija la pestaña History (Historial). Esta página indica cuándo AWS Glue realiza las siguientes tareas:
   - Importar etiquetas
   - Exportar etiquetas
   - Generar etiquetas
   - Estimar calidad
6. Para crear una mejor transformación, puede descargar, etiquetar y cargar de forma iterativa el archivo de etiquetado. En las ejecuciones iniciales, es posible que una gran cantidad de registros no coincidan. Sin embargo, AWS Glue aprende a medida que sigue enseñándole mediante la comprobación del archivo de etiquetado.
7. Evalúe y ajuste su transformación mediante la evaluación del rendimiento y resultados de búsqueda de coincidencias. Para obtener más información, consulte Ajuste de transformaciones de machine learning en AWS Glue.

Etiquetado

Cuando `FindMatches` genera un archivo de etiquetado, los registros se seleccionan en la tabla de origen. En función del entrenamiento anterior, `FindMatches` identifica los registros más valiosos de los que puede aprender.

La acción de etiquetado consiste en editar un archivo (le recomendamos que utilice una hoja de cálculo como Microsoft Excel) y agregar en la columna `label` identificadores o etiquetas que identifiquen los registros coincidentes y no coincidentes. Es importante disponer de una definición clara y coherente de las coincidencias en los datos de origen. `FindMatches` aprende a partir de los registros que se designan (o no) como coincidencias y utiliza estas decisiones para aprender a buscar registros duplicados.

Cuando `FindMatches` genere un archivo de etiquetado, se generarán aproximadamente 100 registros. Estos 100 registros normalmente se dividen en 10 conjuntos de etiquetado, cada uno de los cuales se identifica mediante un `labeling_set_id` único generado por `FindMatches`. Cada conjunto de etiquetado debe considerarse como una tarea de etiquetado independiente de los demás conjuntos de etiquetado. Su tarea consiste en identificar registros coincidentes y no coincidentes dentro de cada conjunto de etiquetado.

Sugerencias para editar archivos de etiquetado en una hoja de cálculo.

Cuando edite el archivo de etiquetado en una aplicación de hoja de cálculo, tenga en cuenta lo siguiente:

- Es posible que el archivo no se abra con los campos de columna totalmente expandidos. Puede que tenga que expandir las columnas `labeling_set_id` y `label` para ver el contenido de esas celdas.
- Si la columna de clave principal es un nombre, como un tipo de datos `long`, es posible que la hoja de cálculo la interprete como un número y cambie el valor. Este valor de la clave debe tratarse como texto. Para solucionar este problema, dé formato a todas las celdas en la columna de clave principal como datos de texto.

Formato de archivo de etiquetado

El archivo de etiquetado que AWS Glue genera para enseñar a su transformación `FindMatches` utiliza el siguiente formato. Si genera su propio archivo de AWS Glue, también debe utilizar este formato:

- Es un archivo de valores separados por comas (CSV).
- Debe estar codificado en `UTF-8`. Si edita el archivo con Microsoft Windows, se puede codificar con `cp1252`.
- Debe ser una ubicación de Amazon S3 para transferirlo a AWS Glue.
- Use una cantidad de filas moderada para cada tarea de etiquetado. Se recomiendan entre 10 y 20 filas por tarea, aunque se considera aceptable utilizar entre 2 y 30 filas. No se recomiendan tareas de más de 50 filas, ya que pueden generar resultados deficientes o errores en el sistema.
- Si dispone de datos que ya están etiquetados y que se componen de pares de registros etiquetados como «coincidencia» o «no coincidencia», puede utilizarlos. Estos pares etiquetados pueden representarse como conjuntos de etiquetado de tamaño 2. En este caso, etiquete los dos registros, por ejemplo, con una letra «A» si coinciden y, si no coinciden, etiquete uno como «A» y otro como «B».

  > **Note**
  >
  > Puesto que dispone de columnas adicionales, el archivo de etiquetado dispone de un esquema diferente de un archivo que contiene sus datos de origen. Coloque el archivo de etiquetado en una carpeta diferente de cualquier archivo CSV de entrada de transformación de manera que el rastreador de AWS Glue no lo tenga en cuenta cuando cree las tablas en el Catálogo de datos. De lo contrario, las tablas creadas por el rastreador de AWS Glue podrían representar incorrectamente sus datos.

- AWS Glue requiere las dos primeras columnas (`labeling_set_id` y `label`). Las columnas restantes deben coincidir con el esquema de los datos que se van a procesar.
- En cada `labeling_set_id`, debe identificar todos los registros coincidentes utilizando la misma etiqueta. Una etiqueta es una cadena única que se incluye en la columna `label`. Le recomendamos utilizar etiquetas que contengan caracteres simples, como A, B, C, etc. Las etiquetas distinguen mayúsculas y minúsculas y se colocan en la columna `label`.
- Se entiende que las filas que contienen el mismo `labeling_set_id` y la misma etiqueta se etiquetan como coincidencia.
- Se entiende que las filas que contienen el mismo `labeling_set_id` y una etiqueta diferente se etiquetan como no coincidencia.
- Se entiende que las filas que contienen un `labeling_set_id` diferente no transmiten información que permita determinar si es o no una coincidencia.

A continuación se muestra un ejemplo de etiquetado de los datos:

labeling_set_id	etiqueta	first_name	last_name	Birthday
ABC123	A	John	Doe	04/01/1980
ABC123	B	Jane	Smith	04/03/1980
ABC123	A	Johnny	Doe	04/01/1980
ABC123	A	Jon	Doe	04/01/1980
DEF345	A	Richard	Jones	12/11/1992
DEF345	A	Rich	Jones	11/12/1992
DEF345	B	Sarah	Jones	12/11/1992
DEF345	C	Richie	Jones Jr.	05/06/2017
DEF345	B	Sarah	Jones-Walker	12/11/1992
GHI678	A	Robert	Miller	1/3/1999
GHI678	A	Bob	Miller	1/3/1999
XYZABC	A	William	Robinson	2/5/2001
XYZABC	B	Andrew	Robinson	2/5/1971

- En el ejemplo anterior identificamos a John/Johnny/Jon Doe como una coincidencia y enseñamos al sistema que estos registros no coinciden con Jane Smith. Por otro lado, enseñamos al sistema que Richard y Rich Jones son la misma persona, pero que estos registros no coinciden con Sarah Jones/Jones-Walker ni Richie Jones Jr.
- Como puede ver, el alcance de las etiquetas está limitado por el `labeling_set_id`. Por lo tanto, las etiquetas no superan los límites de `labeling_set_id`. Por ejemplo, una etiqueta "A" en `labeling_set_id` 1 no tiene ninguna relación con la etiqueta "A" en `labeling_set_id` 2.
- Si un registro no tiene coincidencias dentro de un conjunto de etiquetado, debe asignarle una etiqueta única. Por ejemplo, Jane Smith no coincide con ningún registro del conjunto de etiquetado ABC123, por lo que es el único registro de ese conjunto de etiquetado con la etiqueta B.

- El conjunto de etiquetado «GHI678" demuestra que un conjunto de etiquetado puede componerse exclusivamente de dos registros a los que se les da la misma etiqueta para indicar que coinciden. Del mismo modo, «XYZABC» muestra dos registros con etiquetas diferentes para indicar que no coinciden.
- Tenga en cuenta que a veces los conjuntos de etiquetado pueden no contener coincidencias (es decir, cada registro del conjunto de etiquetado tiene una etiqueta diferente) o tener todos los registros iguales (todos con la misma etiqueta). Esto es correcto siempre que, colectivamente, los conjuntos de etiquetado contengan ejemplos de registros que sean iguales y diferentes, según sus criterios.

> **Important**
>
> Confirme que el rol de IAM que transfiere a AWS Glue tenga acceso al bucket de Amazon S3 que contiene el archivo de etiquetado. Convencionalmente, las políticas de AWS Glue conceden permiso a buckets o carpetas de Amazon S3 cuyos nombres tienen el prefijo aws-glue-. Si sus archivos de etiquetado están en una ubicación distinta, agregue permiso a esa ubicación en el rol de IAM.

Ajuste de transformaciones de machine learning en AWS Glue

Puede ajustar las transformaciones de machine learning en AWS Glue para mejorar los resultados de sus trabajos de limpieza de datos para cumplir sus objetivos. Para mejorar su transformación, puede enseñarle mediante la generación de un conjunto de etiquetas, la adición de etiquetas y, a continuación, la repetición de estos pasos varias veces hasta que disponga de los resultados deseados. También puede realizar el ajuste al modificar algunos parámetros de machine learning.

Para obtener más información sobre las transformaciones de machine learning, consulte Coincidencia de registros con FindMatches de AWS Lake Formation.

Temas

- Mediciones de machine learning
- Decidir entre precisión y exhaustividad
- Decisión entre exactitud y costo
- Estimación de la calidad de coincidencias mediante las puntuaciones de confianza de las coincidencias

- Enseñanza de la transformación de búsqueda de coincidencias

Mediciones de machine learning

Para comprender las mediciones que se utilizan para ajustar la transformación de machine learning, debe estar familiarizado con la siguiente terminología:

Verdadero positivo (VP)

Una coincidencia en los datos que ha encontrado la transformación correctamente. Se denomina a veces "acierto".

Verdadero negativo (VN)

Una no coincidencia en los datos que la transformación ha rechazado correctamente.

Falso positivo (FP)

Una no coincidencia en los datos que la transformación clasificó erróneamente como una coincidencia, que, en ocasiones, se denomina falsa alarma.

Falso negativo (FN)

Una coincidencia en los datos que la transformación no encontró. Se denomina a veces "fallo".

Para obtener más información sobre la terminología que se utiliza en el machine learning, consulte Confusion matrix (Matriz de confusión) en Wikipedia.

Para ajustar sus transformaciones de machine learning, puede cambiar el valor de las siguientes mediciones en la sección de propiedades avanzadas de la transformación.

- Precisión mide qué tan bien la transformación encuentra verdaderos positivos entre el número total de registros que identifica como positivos (verdaderos positivos y falsos positivos). Para obtener más información, consulte Precisión y exhaustividad en Wikipedia.
- La exhaustividad mide la facilidad con la que de la transformación encuentra verdaderos positivos en los registros totales en los datos de origen. Para obtener más información, consulte Precisión y exhaustividad en Wikipedia.
- Laexactitud mide la facilidad con la que la transformación encuentra verdaderos positivos y verdaderos negativos. El aumento de la exactitud requiere más recursos informáticos y costos. Sin embargo, también genera una mayor exhaustividad. Para obtener más información, consulte Precisión y exactitud en Wikipedia.

- El costo mide la cantidad de recursos informáticos y, por lo tanto, dinero, necesario para ejecutar la transformación.

Decidir entre precisión y exhaustividad

Cada transformación`FindMatches` contiene un parámetro `precision-recall`. Utilice este parámetro para especificar una de las siguientes opciones:

- Si está más preocupado por la transformación que indica falsamente que dos registros coinciden cuando realmente no lo hace, debe hacer hincapié en precisión.
- Si está más preocupado por la transformación que no detecta correctamente los registros que coinciden, debe hacer hincapié en exhaustividad.

Puede realizar esta compensación en la consola de AWS Glue o mediante las operaciones de la API de machine learning de AWS Glue.

Cuándo favorecer la precisión

Favorezca la precisión si está más preocupado por el resigo que genera `FindMatches` en un par de registros que coinciden cuando realmente no coinciden. Para favorecer la precisión, elija un valor de compensación de precisión-exhaustividad superior. Con un valor más alto, la transformación `FindMatches` requiere más evidencias para decidir que un par de registros debe coincidir. La transformación se ajusta para sesgarse hacia una situación en la que los registros no coinciden.

Por ejemplo, supongamos que va a utilizar `FindMatches` para detectar elementos duplicados en un catálogo de vídeo y proporciona un valor de precisión-exhaustividad más alto para la transformación. Si su transformación detecta incorrectamente que Star Wars: Una nueva esperanza es igual que Star Wars: El imperio contraataca, es posible que el cliente que quiere Una nueva esperanza vea El imperio contraataca. Esto sería una experiencia del cliente deficiente.

Sin embargo, si la transformación no es capaz de detectar que Star Wars: Una nueva esperanza y Star Wars: Episodio IV - Una nueva esperanza son el mismo elemento, el cliente podría confundirse al principio pero podría reconocerlos como el mismo. Sería un error, pero no tan grave como la situación anterior.

Cuándo favorecer la exhaustividad

Favorezca la exhaustividad si está más preocupado por el riesgo de que se puedan producir errores en los resultados de la transformación `FindMatches` al detectar un par de registros que en

realidad no coinciden. Para favorecer la exhaustividad, elija un valor de compensación de precisión-exhaustividad inferior. Con un valor más bajo, la transformación `FindMatches` requiere menos evidencias para decidir que un par de registros debe coincidir. La transformación se ajusta para sesgarse hacia una situación en la que los registros coinciden.

Por ejemplo, esto podría ser una prioridad para una organización de seguridad. Supongamos que quiere disponer de coincidencias de clientes en relación con una lista de defraudadores conocidos y es importante determinar si un cliente es un defraudador. Está utilizando `FindMatches` para que la lista de defraudadores coincida con la lista de clientes. Cada vez que `FindMatches` detecta una coincidencia entre las dos listas, se asigna un auditor humano para comprobar que la persona es, de hecho, un defraudador. Es posible que su organización prefiera para elegir la exhaustividad en lugar de la precisión. Es decir, puede emplear auditores para que revisen y rechacen manualmente algunos casos cuando el cliente no sea un defraudador en lugar de no poder identificar que un cliente está, de hecho, en la lista de defraudadores.

Cómo favorecer tanto la precisión como la exhaustividad

La mejor forma de mejorar tanto la precisión como la exhaustividad es etiquetar más datos. A medida que etiqueta más datos, mejora la exactitud global de la transformación `FindMatches` y, por lo tanto, tanto la precisión como la exhaustividad son superiores. Sin embargo, incluso con la transformación más precisa, siempre hay un área en gris que necesita experimentar para el favorecimiento de la precisión o la exhaustividad, o elegir un valor intermedio.

Decisión entre exactitud y costo

Cada transformación `FindMatches` contiene un parámetro `accuracy-cost`. Puede utilizar este parámetro para especificar una de las siguientes opciones:

- Si está más preocupado en que la transformación informe de forma exacta de que los dos registros coinciden, debe hacer hincapié en exactitud.
- Si está más preocupado por el costo o la velocidad de ejecución de la transformación, debe hacer hincapié en costo menor.

Puede realizar esta compensación en la consola de AWS Glue o mediante las operaciones de la API de machine learning de AWS Glue.

Cuando favorecer la exactitud

Favorezca la exactitud si está más preocupado por el riesgo de que los resultados de `find matches` no contengan coincidencias. Para favorecer la exactitud, elija un valor de compensación de exactitud-costo superior. Con un volumen más alto, la transformación `FindMatches` requiere más tiempo para realizar una búsqueda más exhaustiva para que los registros coincidan correctamente. Tenga en cuenta que con este parámetro no es menos probable que se trate incorrectamente un par de registro no coincidente como coincidencia. La transformación se ajusta para sesgarse hacia una situación de dedicación de más tiempo a buscar coincidencias.

Cuándo favorecer el costo

Favorezca el costo si está más preocupado por el costo de ejecución de la transformación `find matches` y menos por la cantidad de coincidencias encontradas. Para favorecer el costo, elija un valor de compensación de exactitud-costo inferior. Con un valor inferior, la transformación `FindMatches` requiere menos recursos que ejecutar. La transformación se ajusta para sesgarse hacia una situación de búsqueda de menos coincidencias. Si los resultados son aceptables cuando se favorece el costo inferior, utilice esta configuración.

Cómo favorecer tanto la exactitud como el costo bajo

Es necesario más tiempo de procesamiento para examinar más pares de registros y determinar si deben ser coincidencias. Si desea reducir el costo sin reducir la calidad, estas son algunas medidas que puede adoptar:

- Eliminación de registros en el origen de datos para el que lo le preocupa la coincidencia.
- Eliminación de columnas de origen de datos de las que esté seguro de que no son útiles para crear una decisión de coincidencia/no coincidencia. Una buena forma de decidir esto es eliminar columnas que cree que no afectan a su propia decisión sobre si un conjunto de registros es "el mismo".

Estimación de la calidad de coincidencias mediante las puntuaciones de confianza de las coincidencias

Las puntuaciones de confianza de coincidencias proporcionan una estimación de la calidad de las coincidencias encontradas por FindMatches para distinguir entre registros coincidentes en los que el modelo de machine learning es muy confiable, incierto o improbable. Una puntuación de confianza de coincidencia estará entre 0 y 1, donde una puntuación más alta significa mayor similitud. El análisis de las puntuaciones de confianza de coincidencias le permite distinguir entre grupos de coincidencias en los que el sistema tiene mucha confianza (que puede decidir fusionar), clústeres

sobre los que el sistema no está seguro (que puede decidir hacer revisar por un humano) y clústeres que el sistema considera improbables (que puede decidir rechazar).

Es posible que quiera ajustar sus datos de formación en situaciones en las que vea una puntuación de confianza alta, pero determine que no hay coincidencias, o en las que vea una puntuación baja pero determine que sí hay, de hecho, coincidencias.

Las puntuaciones de confianza son especialmente útiles cuando hay conjuntos de datos industriales de gran tamaño, en los que no es factible revisar todas las decisiones de FindMatches.

Las puntuaciones de confianza de coincidencias están disponibles en AWS Glue, versión 2.0 o posterior.

Generación de puntuaciones de confianza de coincidencias

Puede generar puntuaciones de confianza de coincidencias al establecer el valor booleano de `computeMatchConfidenceScores` a verdadero cuando llama a la API `FindMatches` o `FindIncrementalMatches`.

AWS Glue agrega una nueva `column match_confidence_score` al resultado.

Ejemplos de puntuación de coincidencias

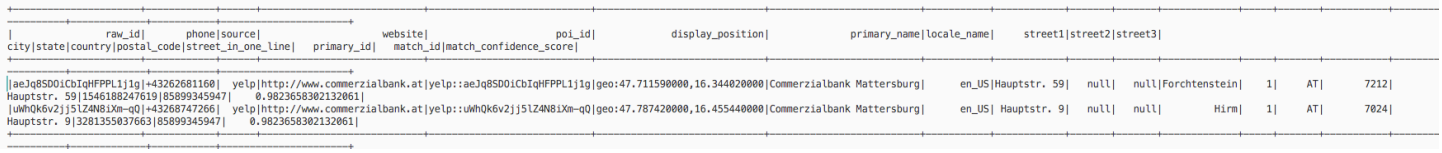
Por ejemplo, considere los siguientes registros coincidentes:

Puntuación >=0,9

Resumen de registros coincidentes:

```
  primary_id  |   match_id  | match_confidence_score

3281355037663    85899345947   0.9823658302132061
1546188247619    85899345947   0.9823658302132061
```

Detalles:



```
+------------+--------------------+------------+------+--------------------+--------------------+--------------------+--------------------+--------------+-------+-------+-------+-------+-------+-------+-------+-------+-------+
|                raw_id|       phone|source|               website|              poi_id|     display_position|        primary_name|locale_name|    street1|street2|street3|
city|state|country|postal_code|street_in_one_line|   primary_id|   match_id|match_confidence_score|
+------------+--------------------+------------+------+--------------------+--------------------+--------------------+--------------------+--------------+-------+-------+-------+-------+-------+-------+-------+-------+-------+
|aeJq8SDOiCbIqHFPPL1j1g|+43262681160|  yelp|http://www.commerzialbank.at|yelp::aeJq8SDOiCbIqHFPPL1j1g|geo:47.711590000,16.344020000|Commerzialbank Mattersburg|      en_US|Hauptstr. 59|   null|   null|Forchtenstein|    1|     AT|       7212|
Hauptstr. 59|1546188247619|85899345947|    0.9823658302132061|
|uWhQk6v2jj5lZ4N8iXm-qQ|+43268747266|  yelp|http://www.commerzialbank.at|yelp::uWhQk6v2jj5lZ4N8iXm-qQ|geo:47.787420000,16.455440000|Commerzialbank Mattersburg|      en_US| Hauptstr. 9|   null|   null|         Hirm|    1|     AT|       7024|
Hauptstr. 9|3281355037663|85899345947|    0.9823658302132061|
+------------+--------------------+------------+------+--------------------+--------------------+--------------------+--------------------+--------------+-------+-------+-------+-------+-------+-------+-------+-------+-------+
```

En este ejemplo, podemos ver que dos registros son muy similares y comparten `display_position`, `primary_name` y `street name`.

Puntuación >=0,8 y puntuación <0,9

Resumen de registros coincidentes:

```
  primary_id  |   match_id  | match_confidence_score

309237680432     85899345928   0.8309852373674638
3590592666790    85899345928   0.8309852373674638
343597390617     85899345928   0.8309852373674638
249108124906     85899345928   0.8309852373674638
463856477937     85899345928   0.8309852373674638
```

Detalles:

raw_id	phone	source	website	poi_id	display_position	primary_name	locale_name	street1	street2	street3	city	state	country	postal_code	street_in_one_line	primary_id	match_id	match_confidence_score
NNIMVA35Tm41mnaokyvr_w	null	yelp	null	yelp::NNIMVA35Tm41mnaokyvr_w	geo:50.541800000,7.102920000	Eiscafe Dolomiten	en_US	Ahrhutstr. 49	null	null	Bad Neuenahr-Ahrweiler	RP	DE	53474	Ahrhutstr. 49	343597390617	85899345928	0.8309852373674638
S3HnQeSvjkc1sht9XQFpeQ	+4956746522	yelp	null	yelp::S3HnQeSvjkc1sht9XQFpeQ	geo:51.447337266,9.414379068	Eiscafé Dolomiten	en_US	Markt 5	null	null	Grebenstein	HE	DE	34393	Markt 5	463856477937	85899345928	0.8309852373674638
o6f-pDXtJmI9PIKpsjx5CQ	+493691744935	yelp	null	yelp::o6f-pDXtJmI9PIKpsjx5CQ	geo:50.976200000,10.324000000	Eiscafe Dolomiten	en_US	Alexanderstr. 105	null	null	Eisenach	TH	DE	99817	Alexanderstr. 105	309237680432	85899345928	0.8309852373674638
D10Q2iYDNXonoG52royfjw	+4926445735	yelp	null	yelp::D10Q2iYDNXonoG52royfjw	geo:50.565900000,7.280050000	Eiscafé Dolomiten	en_US	Rheinstr. 15	null	null	Linz	RP	DE	53545	Rheinstr. 15	3590592666790	85899345928	0.8309852373674638

En este ejemplo, podemos ver que estos registros comparten el mismo `primary_name` y `country`.

Puntuación >=0,6 y puntuación <0,7

Resumen de registros coincidentes:

```
  primary_id  |   match_id  | match_confidence_score

2164663519676    85899345930   0.6971099896480333
 317827595278    85899345930   0.6971099896480333
 472446424341    85899345930   0.6971099896480333
3118146262932    85899345930   0.6971099896480333
 214748380804    85899345930   0.6971099896480333
```

Detalles:

raw_id	phone	source	website	poi_id	display_position	primary_name	locale_name	street1	street2	street3	city	state	country	postal_code	street_in_one_line	primary_id	match_id	match_confidence_score
IQT_R8tk4ngTFXHhpyB8ww	+33490963451	yelp	null	yelp::IQT_R8tk4ngTFXHhpyB8ww	geo:43.675559000,4.626792000	Le Vésuve	en_US	15 Rue de la Rotonde	null	null	Arles	13	FR	13200	15 Rue de la Rotonde	317827595278	85899345930	0.6971099896480333
b8cCAxbvEcug27QMmQYMjQ	null	yelp	null	yelp::b8cCAxbvEcug27QMmQYMjQ	geo:50.631700000,3.067380000	Le Vésuve	en_US	30 ave du President Kennedy	null	null	Lille	59	FR	59800	30 ave du President Kennedy	3118146262932	85899345930	0.6971099896480333
djOC4FZnWXS1wEnFB6vj5g	+33442750804	yelp	null	yelp::djOC4FZnWXS1wEnFB6vj5g	geo:43.427710000,5.236950000	le Vesuve	en_US	24 ave Bruxelles	null	null	Vitrolles	13	FR	13127	24 ave Bruxelles	2164663519676	85899345930	0.6971099896480333
uB59q9Ga561Cljt4wypnkg	+33297251001	yelp	null	yelp::uB59q9Ga561Cljt4wypnkg	geo:48.071493200,-2.963742000	Le Vésuve	en_US	49 Rue Gén de Gaulle	null	null	Pontivy	56	FR	56300	49 Rue Gén de Gaulle	472446424341	85899345930	0.6971099896480333
3wHOMEhra3DUOUgF_YcoTA	+33164069200	yelp	null	yelp::3wHOMEhra3DUOUgF_YcoTA	geo:48.610984000,2.888184000	Le Vesuve	en_US	59 Avenue Charles de Gaulle	null	null	Mormant	77	FR	77720	59 Avenue Charles de Gaulle	214748380804	85899345930	0.6971099896480333

En este ejemplo, podemos ver que estos registros comparten solo el mismo `primary_name`.

Para obtener más información, consulte:

- Paso 5: Agregar y ejecutar un trabajo con su transformación de machine learning.
- PySpark: Clase FindMatches
- PySpark: Clase FindIncrementalMatches
- Scala: Clase FindMatches
- Scala: Clase FindIncrementalMatches

Enseñanza de la transformación de búsqueda de coincidencias

A las transformaciones `FindMatches` se les debe enseñar qué debe considerarse una coincidencia y qué no. Puede enseñar a su transformación mediante la adición de etiquetas en un archivo y la carga de sus opciones a AWS Glue.

Puede organizar esta etiqueta en la consola de AWS Glue o mediante las operaciones de la API de machine learning de AWS Glue.

¿Cuántas veces debo añadir etiquetas? ¿Cuántas etiquetas necesito?

Las respuestas a estas preguntas dependen principalmente de usted. Debe evaluar si `FindMatches` ofrece el nivel de exactitud que necesita y si cree que el esfuerzo de etiquetado adicional merece la pena. La mejor manera de decidir esto es analizar las métricas "Precisión", "Exhaustividad" y "Área bajo la curva de precisión-exhaustividad" que puede generar cuando elije Estimate quality (Estimar calidad) en la consola de AWS Glue. Después de etiquetar más conjuntos de tareas, vuelva a ejecutar esas métricas y compruebe si han mejorado. Si después del etiquetado de algunos conjuntos de tareas, no observa una mejora en la métrica en la que se está centrando, es posible que la calidad de la transformación se haya estabilizado.

¿Por qué las etiquetas de verdadero positivo y verdadero negativo son necesarias?

La transformación `FindMatches` precisa ejemplos positivos y negativos para aprender lo que cree que es una coincidencia. Si está etiquetando datos de entrenamiento generados por `FindMatches` (por ejemplo, mediante la opción I do not have labels [No tengo etiquetas]), `FindMatches` intenta generar un conjunto de "ID de conjuntos de etiquetas" para usted. Dentro de cada tarea, proporcione la misma "etiqueta" a algunos registros y diferentes "etiquetas" a otros. Es decir, por lo general,

las tareas no son todas iguales o todas diferentes (pero es correcto si una tarea determinada es totalmente igual o es totalmente diferente).

Si está enseñando a su transformación `FindMatches` a usar la opción Upload labels from S3 (Cargar etiquetas desde S3), intente incluir ambos ejemplos de registros de coincidencia y no coincidencia. Es aceptable disponer de solo un tipo. Estas etiquetas le ayudan a crear una transformación `FindMatches` más exacta, pero sigue necesitando etiquetas algunos registros que genera mediante la opción Generate labeling file (Generar archivo de etiquetado).

¿Cómo puedo hacer que la transformación coincida exactamente de la forma que le he enseñado?

La transformación `FindMatches` aprende a partir de las etiquetas que proporciona, por lo que puede generar pares de registros que no respeten las etiquetas proporcionadas. Para hacer que la transformación `FindMatches` respete las etiquetas, seleccione EnforceProvidedLabels en FindMatchesParameter.

¿Qué técnicas puede usar cuando una transformación ML identifica elementos como coincidencias que no son coincidencias reales?

Puede utilizar las siguientes técnicas:

- Aumente `precisionRecallTradeoff` a un valor superior. Esto provoca finalmente que se encuentren menos coincidencias, pero también debe fragmentar su clúster de gran tamaño cuando alcance un valor lo suficientemente alto.
- Elija las filas de salida correspondientes a los resultados incorrectos y vuelva a darles formato como conjunto de etiquetas (quitando la columna `match_id` y añadiendo una columna `labeling_set_id` y `label`). Si es necesario, fragmente (subdivida) en varios conjuntos de etiquetas para garantizar que la etiquetadora pueda mantener cada conjunto de etiquetas en cuenta mientras asigna etiquetas. A continuación, etiquete correctamente los conjuntos de coincidencia y cargue el archivo de etiquetas y anéxelo a sus etiquetas existentes. Esto podría enseñar a su transformador lo que necesita sobre lo que está buscando para entender el patrón.
- (Avanzado) Por último, observe los datos para ver si existe un patrón del que el sistema no informa. Preprocese esos datos mediante funciones de AWS Glue estándar para normalizar los datos. Resalte lo que desee que el algoritmo aprenda mediante la separación de datos que sepa que tienen una importancia distinta en sus propias columnas. También puede crear columnas combinadas a partir de columnas con datos que sepa que están relacionados.

Uso de transformaciones de machine learning en la consola de AWS Glue

Puede usarlo AWS Glue para crear transformaciones de aprendizaje automático personalizadas que se pueden usar para limpiar sus datos. Puede usar estas transformaciones cuando cree un trabajo en la consola de AWS Glue .

Para obtener información sobre cómo crear una transformación de machine learning, consulte Coincidencia de registros con FindMatches de AWS Lake Formation.

Temas

- Propiedades de transformación
- Agregado y edición de transformaciones de machine learning
- Visualización de los detalles de la transformación
- Cómo enseñar a las transformaciones mediante el uso de etiquetas

Propiedades de transformación

Para ver una transformación de aprendizaje automático existente, inicia sesión y abre la AWS Glue consola en https://console.aws.amazon.com/glue/. AWS Management Console En el panel de navegación, en Integración de datos y ETL, elija Herramientas de clasificación de datos > Coincidencia de registros.

Las propiedades de cada transformación:

Transform name (Nombre de transformación)

El nombre único que asignó a la transformación cuando la creó.

ID

Un ID único de la transformación.

Recuento de etiquetas

La cantidad de etiquetas en el archivo de etiquetado que se proporcionó para ayudar a enseñar a la transformación.

Status

Indica si la transformación tiene el estado Ready (Listo) o Needs teaching (Necesita formación). Para ejecutar una transformación de machine learning correctamente en un trabajo, el estado debe ser Ready (Listo).

Creado

La fecha en que se creó la transformación.

Modificado

La fecha en la que se actualizó por última vez la transformación.

Descripción

La descripción suministrada para la transformación, si se ha proporcionado una.

Versión de AWS Glue

La versión de AWS Glue usada.

ID de ejecución

El nombre único que asignó a la transformación cuando la creó.

Tipo de tarea

El tipo de transformación de machine learning; por ejemplo, Find matching records (Búsqueda de registros de coincidencia).

Status

Indica el estado de la ejecución de la tarea. Entre las causas posibles se incluyen las siguientes:

- Iniciando
- Running
- Deteniendo
- Stopped (Detenido)
- Succeeded
- Con error
- Tiempo de espera

Error

Si el estado es Con error, aparece un mensaje de error que describe el motivo.

Agregado y edición de transformaciones de machine learning

Puede ver, eliminar, configurar y enseñar, o ajustar una transformación en la consola de AWS Glue. Active la casilla de verificación junto a la transformación en la lista, elija Action (Acción) y, a continuación, seleccione la acción que desee realizar.

Crear una nueva transformación de ML

Para agregar una nueva transformación de machine learning, elija Crear transformación. Siga las instrucciones en el asistente Agregar trabajo. Para obtener más información, consulte Coincidencia de registros con FindMatches de AWS Lake Formation.

Paso 1. Configure propiedades de transformación

1. Ingrese el nombre y la descripción (opcional).
2. Si lo desea, establezca la configuración de seguridad. Consulte Uso de cifrado de datos con transformaciones de machine learning.
3. Si lo desea, establezca la configuración de ejecución de tareas. La configuración de ejecución de tareas permite personalizar la forma en que se ejecuta la tarea. Seleccione el tipo de trabajador, el número de trabajadores, el tiempo de espera de la tarea (en minutos), el número de reintentos y la versión de AWS Glue.
4. Si lo desea, defina las etiquetas. Las etiquetas son etiquetas que se pueden asignar a un AWS recurso. Cada etiqueta consta de una clave y un valor opcional. Las etiquetas se pueden usar para buscar y filtrar el recurso o realizar un seguimiento de AWS los costos.

Paso 2. Elija la tabla y la clave principal.

1. Elija la base de datos y la tabla del catálogo de AWS Glue.
2. Elija una clave principal de la tabla seleccionada. La columna de clave principal normalmente contiene un identificador único para cada registro del origen de datos.

Paso 3. Seleccione las opciones de ajuste.

1. Para Exhaustividad vs. precisión, elija el valor de ajuste para la transformación y así favorecer la exhaustividad o la precisión. De forma predeterminada, está seleccionada la opción Equilibrado, pero puede elegir entre favorecer la exhaustividad o la precisión o bien elegir Personalizado e ingresar un valor entre 0,0 y 1,0 (ambos incluidos).
2. Para menor costo vs. precisión, elija el valor de ajuste que favorezca una reducción del costo o la precisión o bien elija Personalizado e ingrese un valor entre 0,0 y 1,0 (ambos incluidos).
3. Para Forzar la coincidencia, seleccione Forzar la salida para que coincida con las etiquetas si quiere enseñarle a la transformación de ML a que la salida coincida con las etiquetas utilizadas.

Paso 4. Revisar y crear.

1. Revise las opciones de los pasos 1 a 3.
2. Seleccione Editar para cualquier paso que necesite modificarse. Seleccione Crear transformación para completar el asistente de creación de transformaciones.

Uso de cifrado de datos con transformaciones de machine learning

Al agregar una transformación de machine learning a AWS Glue, puede especificar opcionalmente una configuración de seguridad asociada con el origen de datos o el destino de datos. Si el bucket de Amazon S3 utilizado para almacenar los datos está cifrado con una configuración de seguridad, especifique la misma configuración de seguridad al crear la transformación.

También puede optar por utilizar el cifrado del lado del servidor con AWS KMS (SSE-KMS) para cifrar el modelo y las etiquetas y evitar que personas no autorizadas lo inspeccionen. Si eliges esta opción, se te pedirá que elijas AWS KMS key por nombre o puedes elegir Introducir un ARN clave. Si elige ingresar el ARN para la clave KMS, aparecerá un segundo campo en el que puede introducir el ARN de la clave KMS.

> **Note**
>
> En la actualidad, las transformaciones de ML que utilizan una clave de cifrado personalizada no son compatibles en las siguientes regiones:
>
> - Asia-Pacífico (Osaka): `ap-northeast-3`

Visualización de los detalles de la transformación

Visualización de las propiedades de transformación

La página de Propiedades de la transformación incluye los atributos de la transformación. Muestra los detalles sobre la definición de transformación, incluidos los siguientes:

- Transform name (Nombre de transformación) muestra el nombre de la transformación.
- Type (Tipo) muestra el tipo de transformación.
- Status (Estado) muestra si la transformación ya se está usando en un script o trabajo.
- Force output to match labels (Forzar resultado para que las etiquetas coincidan) muestra si la transformación fuerza el resultado para que las etiquetas proporcionadas por el usuario coincidan.

- La Spark version (versión de Spark) se relaciona con la versión de AWS Glue que eligió en Task run properties (Propiedades de ejecución de tarea) al agregar la transformación. Se recomienda AWS Glue 1.0 y Spark 2.4 para la mayoría de los clientes. Para obtener más información, consulte Versiones de AWS Glue.

Pestañas Historial, Estimación de la calidad y Etiquetas

Los detalles de la transformación incluirán la información que definió al crear la transformación. Para ver los detalles de una transformación, seleccione la transformación en la lista Machine learning transforms (Transformaciones de machine learning) y revise la información en las siguientes pestañas:

- Historial
- Estimar calidad
- Etiquetas

Historial

La pestaña History (Historial) muestra su historial de ejecuciones de tareas de transformación. Se ejecutan varios tipos de tareas para enseñar a una transformación. Para cada tarea, las métricas de ejecución incluyen lo siguiente:

- El Run ID (ID de ejecución) es un identificador creado por AWS Glue para cada una de las ejecuciones de esta tarea.
- Task type (Tipo de tarea) muestra el tipo de ejecución de tareas.
- Status (Estado) muestra el éxito de cada tarea que aparece con la ejecución más reciente en la parte superior.
- En Error, se muestran los detalles de un mensaje de error si la ejecución no se ha realizado correctamente.
- En Start time (Hora de inicio) se muestra la fecha y la hora (hora local) en que se inició la tarea.
- En Hora de finalización se muestra la fecha y la hora (hora local) en que finalizó la tarea.
- Logs (Registros) se vincula a los registros escritos en `stdout` para esta ejecución de trabajo.

  El enlace Logs te lleva a Amazon CloudWatch Logs. Allí puede ver los detalles sobre las tablas que se crearon en AWS Glue Data Catalog ellas y los errores encontrados. Puede gestionar el período de retención de registros en la CloudWatch consola. La retención de registros

predeterminada es `Never Expire`. Para obtener más información sobre cómo cambiar el período de retención, consulte Cambiar la retención de datos de registro en los CloudWatch registros en la Guía del usuario de Amazon CloudWatch Logs.

- En Archivo de etiqueta se muestra un enlace a Amazon S3 para un archivo de etiquetado generado.

Estimar calidad

La pestaña Estimate quality (Estimar calidad) muestra las métricas que utiliza para medir la calidad de la transformación. Las estimaciones se calculan comparando las predicciones de coincidencia de transformación mediante un subconjunto de sus datos etiquetados con las etiquetas proporcionadas. Estas estimaciones son aproximadas. Puede invocar una ejecución de tareas de estimación de calidad para esta pestaña.

La pestaña Estimar calidad muestra las métricas de la última ejecución de Estimar calidad incluidas las siguientes propiedades:

- El área bajo la curva Precision-Recall (Precisión-exhaustividad) es un único número que calcula el límite superior de la calidad general de la transformación. Es independiente de la elección realizada para el parámetro precisión-exhaustividad. Los valores más altos indican que cuenta con una compensación de precisión-exhaustividad más atractiva.
- Precision (Precisión) indica la frecuencia con la que la transformación es correcta cuando predice una coincidencia.
- Recall upper limit (Límite superior de exhaustividad) indica la frecuencia con la que la transformación predice la coincidencia en una coincidencia real.
- F1 indica la exactitud de la transformación entre 0 y 1, donde 1 es la mejor exactitud. Para obtener más información, consulte Valor-F en Wikipedia.
- La tabla Column importance (Importancia de columnas) muestra los nombres de columna y la puntuación de importancia para cada columna. La importancia de columna ayuda a comprender cómo contribuyen las columnas al modelo, al identificar qué columnas de los registros se están utilizando con mayor frecuencia para hacer la coincidencia. Estos datos pueden solicitarle que agregue o cambie el conjunto de etiquetas para aumentar o reducir la importancia de las columnas.

  La columna Importance (Importancia) proporciona una puntuación numérica para cada columna, como un decimal no mayor que 1,0.

Para obtener más información sobre cómo comprender las estimaciones de calidad frente a verdadera calidad, consulte Estimaciones de calidad frente a calidad end-to-end (verdadera).

Para obtener más información sobre cómo ajustar su transformación, consulte Ajuste de transformaciones de machine learning en AWS Glue.

Estimaciones de calidad frente a calidad end-to-end (verdadera)

AWS Glue estima la calidad de su transformación al presentar el modelo de machine learning interno con un número de pares de registros que proporcionó para las etiquetas de coincidencia, pero que el modelo no ha visto antes. Estas estimaciones de calidad son una función de la calidad del modelo de machine learning (que está influenciado por el número de registros que etiqueta para "enseñar" a la transformación). El end-to-end recuerdo verdadero (que no es calculado automáticamente por elML `transform`) también está influenciado por el mecanismo de `ML transform` filtrado que propone una amplia variedad de posibles coincidencias con el modelo de aprendizaje automático.

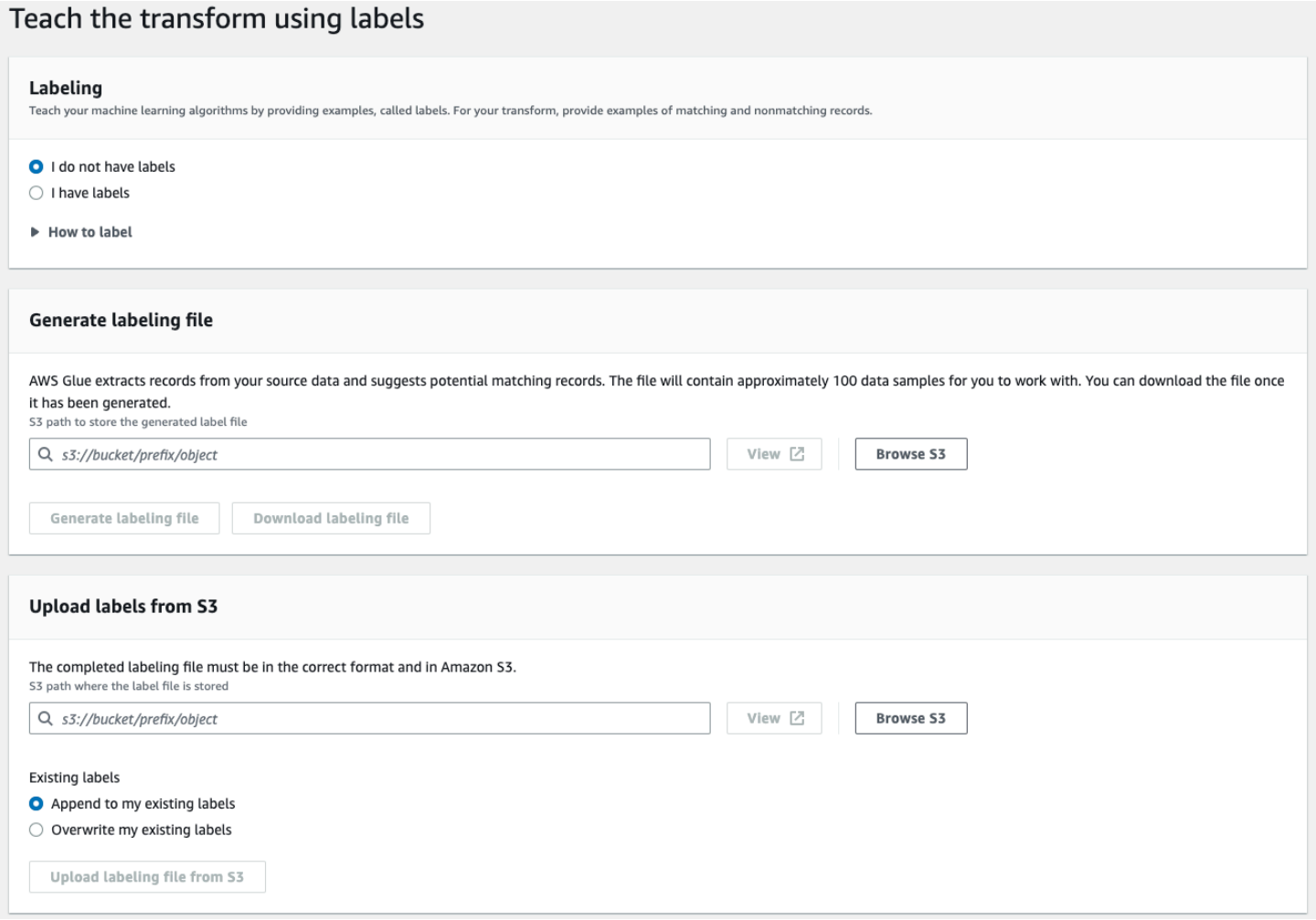
Puede ajustar este método de filtrado principalmente al especificar el valor de ajuste Costo inferior-exactitud. A medida que este valor de ajuste se acerca al valor que favorece la Exactitud, el sistema realiza una búsqueda más completa y más cara de pares de registro que pueden ser coincidencias. Se introducen más pares de registros en su modelo de aprendizaje automático y su `ML transform` memoria verdadera se acerca más a la métrica de recuperación estimada. end-to-end Como resultado, los cambios en la end-to-end calidad de las coincidencias debidos a cambios en la relación coste/precisión de las coincidencias no suelen reflejarse en la estimación de calidad.

Etiquetas

Las etiquetas son etiquetas que puedes asignar a un recurso. AWS Cada etiqueta consta de una clave y un valor opcional. Las etiquetas se pueden usar para buscar y filtrar el recurso o realizar un seguimiento de AWS los costos.

Cómo enseñar a las transformaciones mediante el uso de etiquetas

Puede enseñar a la transformación de ML mediante etiquetas (ejemplos) al seleccionar Enseñar transformación en la página de detalles de la transformación de ML. Si enseña a su algoritmo de machine learning con ejemplos (denominados “etiquetas”), puede elegir las etiquetas existentes para usarlas o crear un archivo de etiquetado.



- Etiquetado: si tiene etiquetas, elija Tengo etiquetas. Si no tiene etiquetas, puede continuar con el siguiente paso, es decir, generar un archivo de etiquetado.
- Generar un archivo de etiquetado: AWS Glue extrae los registros de los datos de origen y sugiere posibles registros de coincidencias. Usted elige el bucket de Amazon S3 para almacenar el archivo de etiquetas generado. Elija Generar archivo de etiquetado para iniciar el proceso. Cuando haya terminado, elija Descargar archivo de etiquetado. El archivo descargado tendrá una columna de etiquetas en la que podrá rellenar las etiquetas.
- Cargar etiquetas desde Amazon S3: elija el archivo de etiquetado completo del bucket de Amazon S3 en el que está almacenado el archivo de etiquetas. A continuación, elija agregar las etiquetas a las etiquetas existentes o sobrescribirlas. Seleccione Cargar archivo de etiquetado desde Amazon S3.

Tutorial: creación de una transformación de machine learning con AWS Glue

Este tutorial le guiará a través de las acciones para crear y administrar una transformación de machine learning (ML) con AWS Glue. Antes de utilizar este tutorial, debe estar familiarizado con el uso de la consola de AWS Glue para añadir rastreadores y trabajos, y editar scripts. También debe estar familiarizado con la búsqueda y descarga de archivos en la consola de Amazon Simple Storage Service (Amazon S3).

En este ejemplo, se creará una transformación `FindMatches` para encontrar los registros coincidentes, enseñarle cómo identificar registros de coincidencia y no coincidencia, y utilizarla en un trabajo de AWS Glue. El trabajo de AWS Glue escribe un nuevo archivo de Amazon S3 con una columna adicional denominada `match_id`.

En este tutorial se ha utilizado un archivo denominado `dblp_acm_records.csv` como dato de origen. Este archivo es una versión modificada de publicaciones académicas (DBLP y ACM) disponible a partir del conjunto de datos de DBLP y ACM original. El archivo `dblp_acm_records.csv` es un archivo de valores separados por comas (CSV) en formato UTF-8 sin marca de orden de bytes (BOM).

El segundo archivo `dblp_acm_labels.csv`, es un archivo de etiquetado de ejemplo que contiene registros de coincidencia y no coincidencia utilizados para enseñar a la transformación como parte del tutorial.

Temas

- Paso 1: Rastrear los datos de origen
- Paso 2: Agregar una transformación de machine learning
- Paso 3: Agregar una transformación de machine learning
- Paso 4: Calcular la calidad de su transformación de machine learning
- Paso 5: Agregar y ejecutar un trabajo con su transformación de machine learning.
- Paso 6: Verificar los datos de salida desde Amazon S3

Paso 1: Rastrear los datos de origen

En primer lugar, rastree el archivo CSV de Amazon S3 de origen para crear una tabla de metadatos correspondiente en el Data Catalog.

> **Important**
>
> Para dirigir el rastreador para crear una tabla únicamente para el archivo CSV, almacene los datos de origen de CSV en una carpeta de Amazon S3 diferente de los demás archivos.

1. Inicie sesión en la AWS Management Console y abra la consola de AWS Glue en https://console.aws.amazon.com/glue/.
2. En el panel de navegación, elija Crawlers (Rastreadores) y Add crawler (Añadir rastreador).
3. Siga el asistente para crear y ejecutar un rastreador denominado `demo-crawl-dblp-acm` con salida a la base de datos `demo-db-dblp-acm`. Cuando ejecute el asistente, cree la base de datos `demo-db-dblp-acm` si no existe todavía. Elija una ruta de inclusión de Amazon S3 a los datos de ejemplo en la región de AWS actual. Por ejemplo, en el caso de `us-east-1`, la ruta de inclusión de Amazon S3 para el archivo de origen es `s3://ml-transforms-public-datasets-us-east-1/dblp-acm/records/dblp_acm_records.csv`.

   Si se ejecuta correctamente, el rastreador crea la tabla `dblp_acm_records_csv` con las siguientes columnas: id, title (título), authors (autores), venue (lugar), year (año) y source (origen).

Paso 2: Agregar una transformación de machine learning

A continuación, agregue una transformación de machine learning que se base en el esquema de la tabla de origen de datos creada por el rastreador denominada `demo-crawl-dblp-acm`.

1. En la consola de AWS Glue, en el panel de navegación, en la sección Integración de datos y ETL, seleccione Herramientas de clasificación de datos > Coincidencia de registros y, a continuación, Agregar transformación. Siga el asistente para crear una transformación `Find matches` con las siguientes propiedades.

   a. En Transform name (Nombre de transformación), escriba **`demo-xform-dblp-acm`**. Este es el nombre de la transformación que se utiliza para buscar coincidencias en los datos de origen.

   b. En IAM role (Rol de IAM) elija un rol de IAM que tenga permiso para los datos de origen de Amazon S3, el archivo de etiquetado y las operaciones de la API de AWS Glue. Para obtener más información, consulte Creación de un rol de IAM para AWS Glue en la Guía para desarrolladores de AWS Glue.

c. En Data source (Origen de datos), elija la tabla denominada dblp_acm_records_csv en la base de datos demo-db-dblp-acm.

d. En Primary key (Clave principal), elija la columna de clave principal para la tabla, id.

2. En el asistente, elija Finish (Finalizar) y vuelva a la lista ML transforms (Transformaciones de ML).

Paso 3: Agregar una transformación de machine learning

A continuación, debe enseñar a su transformación de machine learning a usar el archivo de etiquetado de ejemplo del tutorial.

No puede utilizar una transformación de lenguaje automático en un trabajo de extracción, transformación y carga (ETL) hasta que el estado sea Ready for use (Listo para su uso). Para que la transformación esté preparada, debe enseñarle cómo identificar registros de coincidencia y no coincidencia mediante el ofrecimiento de ejemplos de registros de coincidencia y no coincidencia. Para enseñar a su transformación, puede generar un archivo de etiquetas, añadir etiquetas y, a continuación, subir un archivo de etiquetas. En este tutorial, puede utilizar el archivo de etiquetado de ejemplo denominado `dblp_acm_labels.csv`. Para obtener más información sobre el proceso de etiquetado, consulte Etiquetado.

1. En el panel de navegación de la consola de AWS Glue, elija Coincidencia de registros.
2. Elija la transformación `demo-xform-dblp-acm` y, a continuación, elija Action (Acción) y Teach (Enseñar). Siga el asistente para enseñar a su transformación `Find matches`.
3. En la página de propiedades de transformación, elija I have labels (Tengo etiquetas). Elija una ruta de Amazon S3 al archivo de etiquetado de ejemplo en la región de AWS actual. Por ejemplo, para `us-east-1`, cargue el archivo de etiquetado proporcionado de la ruta de Amazon S3 `s3://ml-transforms-public-datasets-us-east-1/dblp-acm/labels/dblp_acm_labels.csv` con la opción overwrite (sobreescribir) las etiquetas existentes. El archivo de etiquetado debe estar ubicado en Amazon S3 en la misma región que la consola de AWS Glue.

   Al cargar un archivo de etiquetado, se inicia una tarea en AWS Glue para añadir o sobrescribir las etiquetas utilizadas para enseñar a la transformación cómo procesar el origen de datos.
4. En la última página del asistente, elija Finish (Finalizar), y vuelva a la lista ML transforms (Transformaciones de ML).

Paso 4: Calcular la calidad de su transformación de machine learning

A continuación, puede estimar la calidad de su transformación de machine learning. La calidad depende de la cantidad de etiquetado que acaba de realizar. Para obtener más información sobre la calidad de estimación, consulte Estimar calidad.

1. En la consola de AWS Glue, en el panel de navegación, en Integración de datos y ETL, seleccione Herramientas de clasificación de datos > Coincidencia de registros.
2. Elija la transformación `demo-xform-dblp-acm` y elija la pestaña Estimate quality (Estimar calidad). Esta pestaña muestra las estimaciones de calidad actuales, si estuvieran disponibles, para la transformación.
3. Elija Estimate quality (Estimar calidad) para iniciar una tarea para estimar la calidad de la transformación. La exactitud de la estimación de calidad se basa en el etiquetado de los datos de origen.
4. Diríjase a la pestaña History (Historial). En este panel, aparecen las ejecuciones de tareas para la transformación, incluida la tarea de Estimating quality (Estimación de calidad). Para obtener más información sobre la ejecución, elija Logs (Registros). Compruebe que el estado de ejecución sea Suceeded (Correcto) cuando finalice.

Paso 5: Agregar y ejecutar un trabajo con su transformación de machine learning.

En este paso, utilice la transformación de machine learning para agregar y ejecutar un trabajo en AWS Glue. Cuando la transformación `demo-xform-dblp-acm` esté lista para su uso, podrá utilizarla en un trabajo de ETL.

1. En el panel de navegación de la consola de AWS Glue, seleccione Jobs (Trabajos).
2. Elija Add job (Añadir trabajo) y siga los pasos en el asistente para crear un trabajo de ETL Spark con un script generado. Elija los siguientes valores de propiedad para su transformación:
   a. En Name (Nombre), elija el trabajo de ejemplo en este tutorial, demo-etl-dblp-acm.
   b. En IAM role (Rol de IAM), elija un rol de IAM con permiso para los datos de origen de Amazon S3, el archivo de etiquetado y las operaciones de la API de AWS Glue. Para obtener más información, consulte Creación de un rol de IAM para AWS Glue en la Guía para desarrolladores de AWS Glue.
   c. En ETL language (Lenguaje de ETL), elija Scala. Este es el lenguaje de programación en el script de ETL.

d. En Script file name (Nombre de archivo de script), elija demo-etl-dblp-acm. Este es el nombre de archivo del script de Scala (mismo nombre de archivo).

e. En Data source (Origen de datos), elija dblp_acm_records_csv. El origen de datos que elija debe coincidir con el esquema de origen de datos de transformación de machine learning.

f. En Transform type (Tipo de transformación), elija Find matching records (Buscar registros de coincidencia) para crear un trabajo mediante una transformación de machine learning.

g. Borre Remove duplicate records (Eliminar registros duplicados). No quiere eliminar registros duplicados porque los registros de salida escritos disponen de un campo `match_id` adicional añadido.

h. En Transform (Transformación), elija demo-xform-dblp-acm, la transformación de machine learning utilizada por el trabajo.

i. En Create tables in your data target (Crear tablas en su destino de datos), elija crear tablas con las siguientes propiedades:

- Data store type (Tipo de almacén de datos): **`Amazon S3`**
- Format (Formato): **`CSV`**
- Compression type (Tipo de compresión): **`None`**
- Target path (Ruta de destino): la ruta de Amazon S3 donde se escribe la salida del trabajo (en la región de AWS de la consola actual)

3. Elija Save job and edit script (Guardar trabajo y editar script) para mostrar la página del editor de scripts.

4. Edite el script para añadir una instrucción para que la salida del trabajo en Target path (Ruta de destino) se sobrescriba en un archivo de partición único. Añada esta instrucción inmediatamente después de la instrucción que ejecuta la transformación `FindMatches`. La instrucción es similar a la siguiente.

```
val single_partition = findmatches1.repartition(1)
```

Debe modificar la instrucción `.writeDynamicFrame(findmatches1)` para escribir la salida como `.writeDynamicFrame(single_partion)`.

5. Después de editar el script, elija Save (Guardar). El script modificado tiene un aspecto similar al siguiente código, pero se personaliza para su entorno.

```
import com.amazonaws.services.glue.GlueContext
```

```
import com.amazonaws.services.glue.errors.CallSite
import com.amazonaws.services.glue.ml.FindMatches
import com.amazonaws.services.glue.util.GlueArgParser
import com.amazonaws.services.glue.util.Job
import com.amazonaws.services.glue.util.JsonOptions
import org.apache.spark.SparkContext
import scala.collection.JavaConverters._

object GlueApp {
  def main(sysArgs: Array[String]) {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)
    // @params: [JOB_NAME]
    val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB_NAME").toArray)
    Job.init(args("JOB_NAME"), glueContext, args.asJava)
    // @type: DataSource
    // @args: [database = "demo-db-dblp-acm", table_name = "dblp_acm_records_csv",
 transformation_ctx = "datasource0"]
    // @return: datasource0
    // @inputs: []
    val datasource0 = glueContext.getCatalogSource(database = "demo-db-dblp-acm",
 tableName = "dblp_acm_records_csv", redshiftTmpDir = "", transformationContext =
 "datasource0").getDynamicFrame()
    // @type: FindMatches
    // @args: [transformId = "tfm-123456789012", emitFusion = false,
 survivorComparisonField = "<primary_id>", transformation_ctx = "findmatches1"]
    // @return: findmatches1
    // @inputs: [frame = datasource0]
    val findmatches1 = FindMatches.apply(frame = datasource0, transformId
 = "tfm-123456789012", transformationContext = "findmatches1",
 computeMatchConfidenceScores = true)


    // Repartition the previous DynamicFrame into a single partition.
    val single_partition = findmatches1.repartition(1)


    // @type: DataSink
    // @args: [connection_type = "s3", connection_options = {"path": "s3://aws-
glue-ml-transforms-data/sal"}, format = "csv", transformation_ctx = "datasink2"]
    // @return: datasink2
    // @inputs: [frame = findmatches1]
    val datasink2 = glueContext.getSinkWithFormat(connectionType =
 "s3", options = JsonOptions("""{"path": "s3://aws-glue-ml-transforms-
```

```
data/sal"}"""), transformationContext = "datasink2", format =
 "csv").writeDynamicFrame(single_partition)
    Job.commit()
  }
}
```

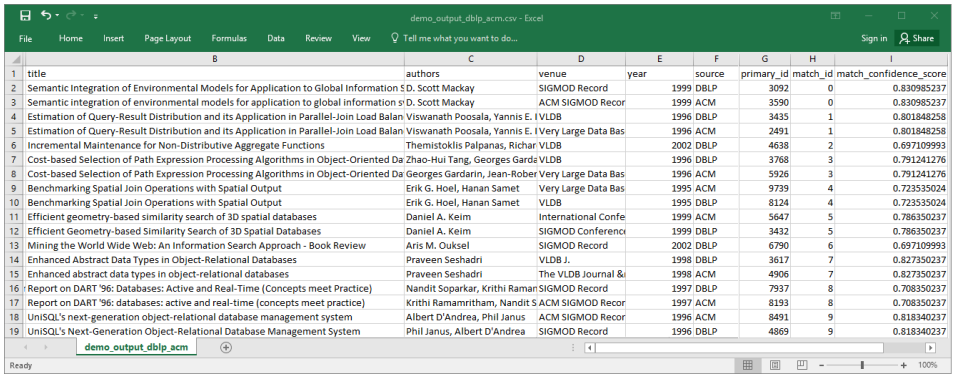
6. Elija Run job (Ejecutar trabajo) para iniciar la ejecución de trabajo. Compruebe el estado del trabajo en la lista de trabajos. Cuando el trabajo finaliza, en ML transform (Transformación ML), en la pestaña History (Historial), habrá una nueva fila Run ID (ID de ejecución) añadida al tipo ETL job (Trabajo de ETL).
7. Diríjase a Jobs (Trabajos), pestaña History (Historial). En este panel, aparecen las ejecuciones de trabajo. Para obtener más información sobre la ejecución, elija Logs (Registros). Compruebe que el estado de ejecución sea Suceeded (Correcto) cuando finalice.

Paso 6: Verificar los datos de salida desde Amazon S3

En este paso, verificará la salida de la ejecución de trabajo en el bucket de Amazon S3 que elija cuando agregue el trabajo. Puede descargar el archivo de salida en su máquina local y verificar que se identificaron los registros de coincidencia.

1. Abra la consola de Amazon S3 en https://console.aws.amazon.com/s3/.
2. Descargue el archivo de salida de destino del trabajo `demo-etl-dblp-acm`. Abra el archivo en una aplicación de hoja de cálculo (es posible que tenga que añadir una extensión de archivo `.csv` para que el archivo se abra correctamente).

   La imagen siguiente muestra un fragmento de la salida en Microsoft Excel.



	B	C	D	E	F	G	H	I
1	title	authors	venue	year	source	primary_id	match_id	match_confidence_score
2	Semantic Integration of Environmental Models for Application to Global Information S	D. Scott Mackay	SIGMOD Record	1999	DBLP	3092	0	0.830985237
3	Semantic integration of environmental models for application to global information s	D. Scott Mackay	ACM SIGMOD Recor	1999	ACM	3590	0	0.830985237
4	Estimation of Query-Result Distribution and its Application in Parallel-Join Load Balan	Viswanath Poosala, Yannis E. I	VLDB	1996	DBLP	3435	1	0.801848258
5	Estimation of Query-Result Distribution and its Application in Parallel-Join Load Balan	Viswanath Poosala, Yannis E. I	Very Large Data Bas	1996	ACM	2491	1	0.801848258
6	Incremental Maintenance for Non-Distributive Aggregate Functions	Themistoklis Palpanas, Richar	VLDB	2002	DBLP	4638	2	0.697109993
7	Cost-based Selection of Path Expression Processing Algorithms in Object-Oriented Da	Zhao-Hui Tang, Georges Garda	VLDB	1996	DBLP	3768	3	0.791241276
8	Cost-based Selection of Path Expression Processing Algorithms in Object-Oriented Da	Georges Gardarin, Jean-Rober	Very Large Data Bas	1996	ACM	5926	3	0.791241276
9	Benchmarking Spatial Join Operations with Spatial Output	Erik G. Hoel, Hanan Samet	Very Large Data Bas	1995	ACM	9739	4	0.723535024
10	Benchmarking Spatial Join Operations with Spatial Output	Erik G. Hoel, Hanan Samet	VLDB	1995	DBLP	8124	4	0.723535024
11	Efficient geometry-based similarity search of 3D spatial databases	Daniel A. Keim	International Confe	1999	ACM	5647	5	0.786350237
12	Efficient Geometry-based Similarity Search of 3D Spatial Databases	Daniel A. Keim	SIGMOD Conferenc	1999	DBLP	3432	5	0.786350237
13	Mining the World Wide Web: An Information Search Approach - Book Review	Aris M. Ouksel	SIGMOD Record	2002	DBLP	6790	6	0.697109993
14	Enhanced Abstract Data Types in Object-Relational Databases	Praveen Seshadri	VLDB J.	1998	DBLP	3617	7	0.827350237
15	Enhanced abstract data types in object-relational databases	Praveen Seshadri	The VLDB Journal &	1998	ACM	4906	7	0.827350237
16	Report on DART '96: Databases: Active and Real-Time (Concepts meet Practice)	Nandit Soparkar, Krithi Raman	SIGMOD Record	1997	DBLP	7937	8	0.708350237
17	Report on DART '96: databases: active and real-time (concepts meet practice)	Krithi Ramamritham, Nandit S	ACM SIGMOD Recor	1997	ACM	8193	8	0.708350237
18	UniSQL's next-generation object-relational database management system	Albert D'Andrea, Phil Janus	ACM SIGMOD Recor	1996	ACM	8491	9	0.818340237
19	UniSQL's Next-Generation Object-Relational Database Management System	Phil Janus, Albert D'Andrea	SIGMOD Record	1996	DBLP	4869	9	0.818340237

   El archivo de origen y destino de datos tiene 4911 registros. Sin embargo, la transformación `Find matches` añade otra columna denominada `match_id` para identificar registros de coincidencia en la salida. Las filas con el mismo `match_id` se consideran registros de

coincidencia. La `match_confidence_score` es un número entre 0 y 1 que proporciona una estimación de la calidad de las coincidencias encontradas por `Find matches`.

3. Ordene el archivo de salida por `match_id` para ver fácilmente qué registros son coincidencias. Compare los valores en las demás columnas para ver si acepta los resultados de la transformación `Find matches`. Si no es así, puede seguir enseñando a la transformación mediante la adición de más etiquetas.

   También puede ordenar el archivo por otro campo, como `title`, para ver si los registros con títulos similares tienen el mismo `match_id`.

Búsqueda de coincidencias progresivas

La característica de búsqueda de coincidencias permite identificar registros duplicados o coincidentes en el conjunto de datos, incluso cuando los registros no tienen un identificador único común y no coinciden exactamente los campos. La versión inicial de búsqueda de coincidencia transforma los registros coincidentes identificados dentro de un único conjunto de datos. Cuando agregó datos nuevos al conjunto de datos, tuvo que fusionarlo con el conjunto de datos limpio existente y volver a ejecutar la coincidencia con el conjunto de datos fusionado completo.

La característica de coincidencia progresiva facilita la coincidencia con los registros progresivos con respecto a los conjuntos de datos coincidentes existentes. Suponga que desea asociar los datos de los clientes potenciales con los conjuntos de datos de clientes existentes. La capacidad de coincidencia progresiva proporciona la flexibilidad de asociar cientos de miles de nuevos clientes potenciales con una base de datos existente de clientes potenciales y clientes existentes mediante la fusión de los resultados en una única base de datos o tabla. Al hacer coincidir solo entre los conjuntos de datos nuevos y existentes, la optimización de búsqueda de coincidencias progresivas reduce el tiempo de cálculo, lo que también reduce los costos.

La utilización de la coincidencia progresiva es similar a la búsqueda de coincidencias tal como se describe en Tutorial: creación de una transformación de machine learning con AWS Glue. En este tema, se identifican únicamente las diferencias con la coincidencia progresiva.

Para obtener más información, consulte la publicación del blog de Coincidencia progresiva de datos.

Ejecución de un trabajo de coincidencia progresiva

Para el siguiente procedimiento, suponga lo siguiente:

- Se ha rastreado el conjunto de datos existente y los resultados se han pasado a la tabla first_records. El conjunto de datos de first_records debe ser un conjunto de datos coincidente, o bien el resultado del trabajo coincidente.
- Se ha creado y entrenado una transformación de FindMatches con AWS Glue versión 2.0. Esta es la única versión de AWS Glue que soporta coincidencias progresivas.
- El lenguaje de ETL es Scala. Tenga en cuenta que también se soporta Python.
- El modelo ya generado se denomina `demo-xform`.

1. Rastree el conjunto de datos progresivo hasta la tabla second_records.
2. En el panel de navegación de la consola de AWS Glue, seleccione Jobs (Trabajos).
3. Elija Add job (Añadir trabajo) y siga los pasos en el asistente para crear un trabajo de ETL Spark con un script generado. Elija los siguientes valores de propiedad para su transformación:
   a. Para Name (Nombre), elija demo-etl.
   b. En IAM role (Rol de IAM), elija un rol de IAM con permiso para los datos de origen de Amazon S3, el archivo de etiquetado y las operaciones de la API de AWS Glue.
   c. En ETL language (Lenguaje de ETL), elija Scala.
   d. En Script file name (Nombre de archivo de script), elija demo-etl. Este es el nombre de archivo del script de Scala.
   e. Para Data source (Origen de datos), elija first_records. El origen de datos que elija debe coincidir con el esquema de origen de datos de transformación de machine learning.
   f. En Transform type (Tipo de transformación), elija Find matching records (Buscar registros de coincidencia) para crear un trabajo mediante una transformación de machine learning.
   g. Seleccione la opción de coincidencia progresiva y para Data source (Origen de datos), seleccione la tabla denominada second_records.
   h. En Transform (Transformación), elija demo-xform, la transformación de machine learning utilizada por el trabajo.
   i. Elija Create tables in your data target (Crear tablas en el destino de datos) o Use tables in the data catalog and update your data target (Utilizar tablas en el Catálogo de datos y actualizar el destino de datos).
4. Elija Save job and edit script (Guardar trabajo y editar script) para mostrar la página del editor de scripts.
5. Elija Run job (Ejecutar trabajo) para iniciar la ejecución de trabajo.

Uso de FindMatches en un trabajo visual

Para usar la transformación FindMatches en AWS Glue Studio, puede usar el nodo Custom Transform que invoca la API FindMatches. Para obtener más información sobre cómo usar una transformación personalizada, consulte Crear una transformación personalizada.

> **Note**
>
> Actualmente, la API FindMatches solo funciona con `Glue 2.0`. Para ejecutar un trabajo con la transformación personalizada que invoca la API FindMatches, asegúrese de que la versión de AWS Glue sea `Glue 2.0` en la pestaña Detalles del trabajo. Si la versión de AWS Glue no es `Glue 2.0`, el trabajo fallará en tiempo de ejecución y mostrará el siguiente mensaje de error: “no se puede importar el nombre 'FindMatches' de 'awsglueml.transforms'”.

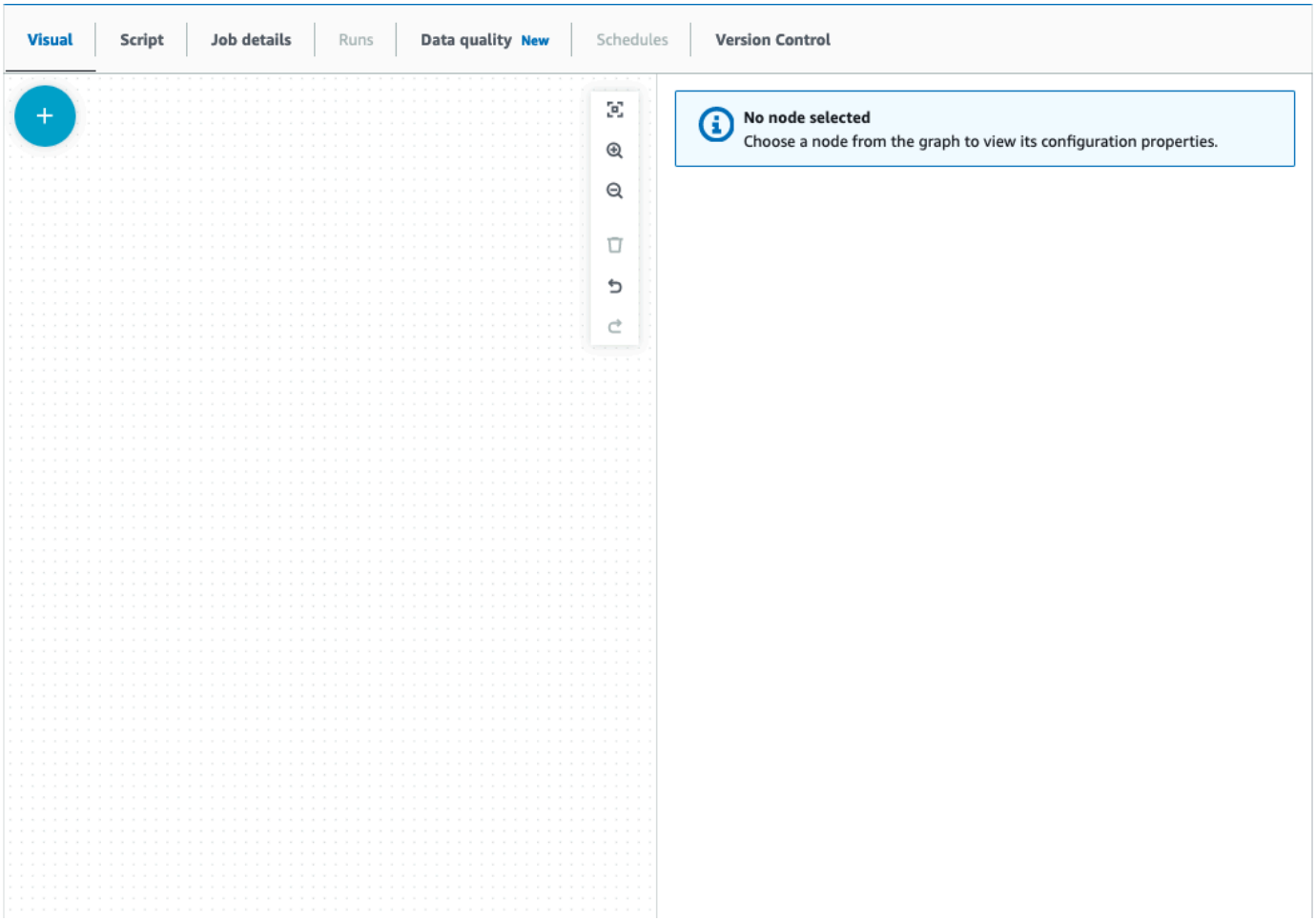
Requisitos previos

- Para utilizar la transformación FindMatches abra la consola de AWS Glue Studio en https://console.aws.amazon.com/gluestudio/.
- Cree una transformación de machine learning. Cuando se crea, se genera un transformId. Para realizar los pasos que se indican a continuación, necesitará este ID. Para obtener más información sobre cómo crear una transformación de machine learning, consulte Agregar y editar transformaciones de machine learning.

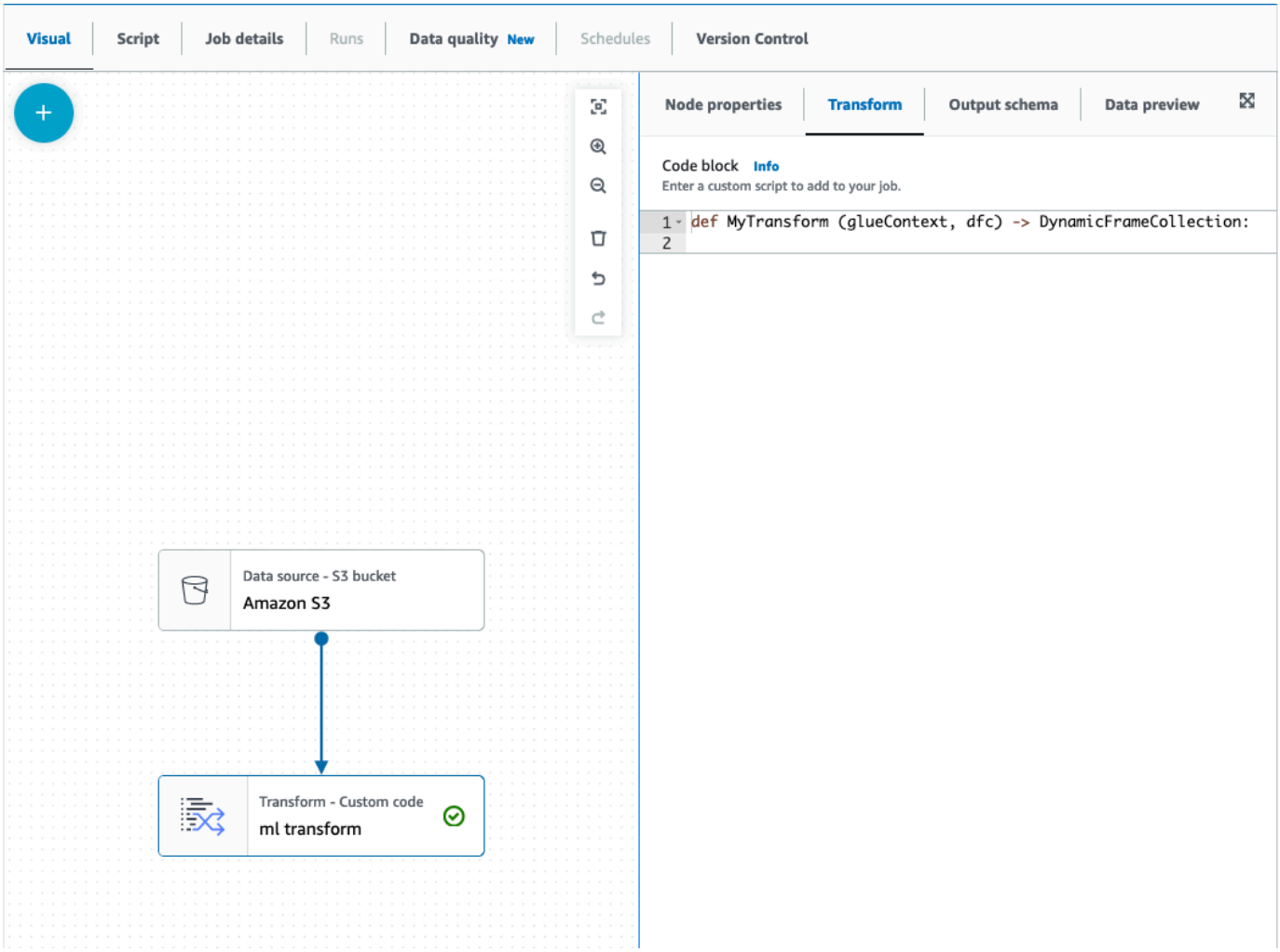
Agregar una transformación de FindMatches

Para agregar una transformación de FindMatches:

1. En el editor de trabajos de AWS Glue Studio, abra el panel de recursos mediante un clic en el símbolo de la cruz situado en la esquina superior izquierda del gráfico visual del trabajo y elija un origen de datos en la pestaña Datos. Este es el origen de datos en el que desea comprobar si hay coincidencias.



2. Seleccione el nodo del origen de datos y, a continuación, abra el panel de recursos mediante un clic en el símbolo de la cruz situado en la esquina superior izquierda del gráfico visual del trabajo y busque “transformación personalizada”. Elija el nodo Transformación personalizada para agregarlo al gráfico. La transformación personalizada está vinculada al nodo de origen de datos. Si no lo está, puede hacer clic en el nodo Transformación personalizada y elegir la pestaña Propiedades del nodo y, a continuación, en Nodos principales, elegir el origen de datos.
3. Haga clic en el nodo Transformación personalizada del gráfico visual y, a continuación, seleccione la pestaña Propiedades del nodo y asigne un nombre a la transformación personalizada. Se recomienda cambiar el nombre de la transformación para que el nombre de la transformación sea fácilmente identificable en el gráfico visual.
4. Seleccione la pestaña Transformación, donde podrá editar el bloque de código. Aquí es donde se puede agregar el código para invocar la API FindMatches.



El bloque de código contiene código rellenado previamente para que pueda empezar. Sobrescriba el código rellenado previamente con la plantilla que se muestra a continuación. La plantilla tiene un marcador de posición para el transformId, que puede proporcionar.

```
def MyTransform (glueContext, dfc) -> DynamicFrameCollection:
    dynf = dfc.select(list(dfc.keys())[0])
    from awsglueml.transforms import FindMatches
    findmatches = FindMatches.apply(frame = dynf, transformId = "<your id>")
    return(DynamicFrameCollection({"FindMatches": findmatches}, glueContext))
```

5. Haga clic en el nodo Transformación personalizada en el gráfico visual y, a continuación, abra el panel de recursos mediante un clic en el símbolo de la cruz situado en la esquina superior izquierda del gráfico visual del trabajo y busque “Seleccionar de la colección”. No es necesario cambiar la selección predeterminada, ya que solo hay un DynamicFrame en la colección.

6. Puede seguir agregando transformaciones o almacenar el resultado, que ahora está enriquecido con las columnas adicionales de búsqueda de coincidencias. Si desea hacer referencia a esas nuevas columnas en las transformaciones posteriores, debe agregarlas al esquema de salida de la transformación. La forma más sencilla de hacerlo es elegir la pestaña de vista previa de datos y, a continuación, en la pestaña de esquema, elegir “Usar esquema de vista previa de datos”.
7. Para personalizar FindMatches, puede agregar parámetros adicionales para pasarlos al método 'aplicar'. Consulte Clase FindMatches.

Agregar una transformación de FindMatches de forma incremental

En el caso de coincidencias incrementales, el proceso es el mismo que el de Agregar una transformación FindMatches con las siguientes diferencias:

- En lugar de un nodo principal para la transformación personalizada, necesita dos nodos principales.
- El primer nodo principal debe ser el conjunto de datos.
- El segundo nodo principal debe ser el conjunto de datos incremental.

  Sustituya `transformId` con su `transformId` en la plantilla de bloque de código:

```
def MyTransform (glueContext, dfc) -> DynamicFrameCollection:
    dfs = list(dfc.values())
    dynf = dfs[0]
    inc_dynf = dfs[1]
    from awsglueml.transforms import FindIncrementalMatches
    findmatches = FindIncrementalMatches.apply(existingFrame = dynf, incrementalFrame
 = inc_dynf,
                                    transformId = "<your id>")
    return(DynamicFrameCollection({"FindMatches": findmatches}, glueContext))
```

- Para ver los parámetros opcionales, consulte Clase FindIncrementalMatches.

Migrar programas de Apache Spark a AWS Glue

Apache Spark es una plataforma de código abierto para cargas de trabajo de computación distribuidas que se realizan en conjuntos de datos de gran tamaño. AWS Glue aprovecha las capacidades de Spark para ofrecer una experiencia de ETL optimizada. Puede migrar programas

de Spark a AWS Glue para aprovechar nuestras características. AWS Glue proporciona las mismas mejoras de rendimiento que cabría esperar de Apache Spark en Amazon EMR.

Ejecutar código Spark

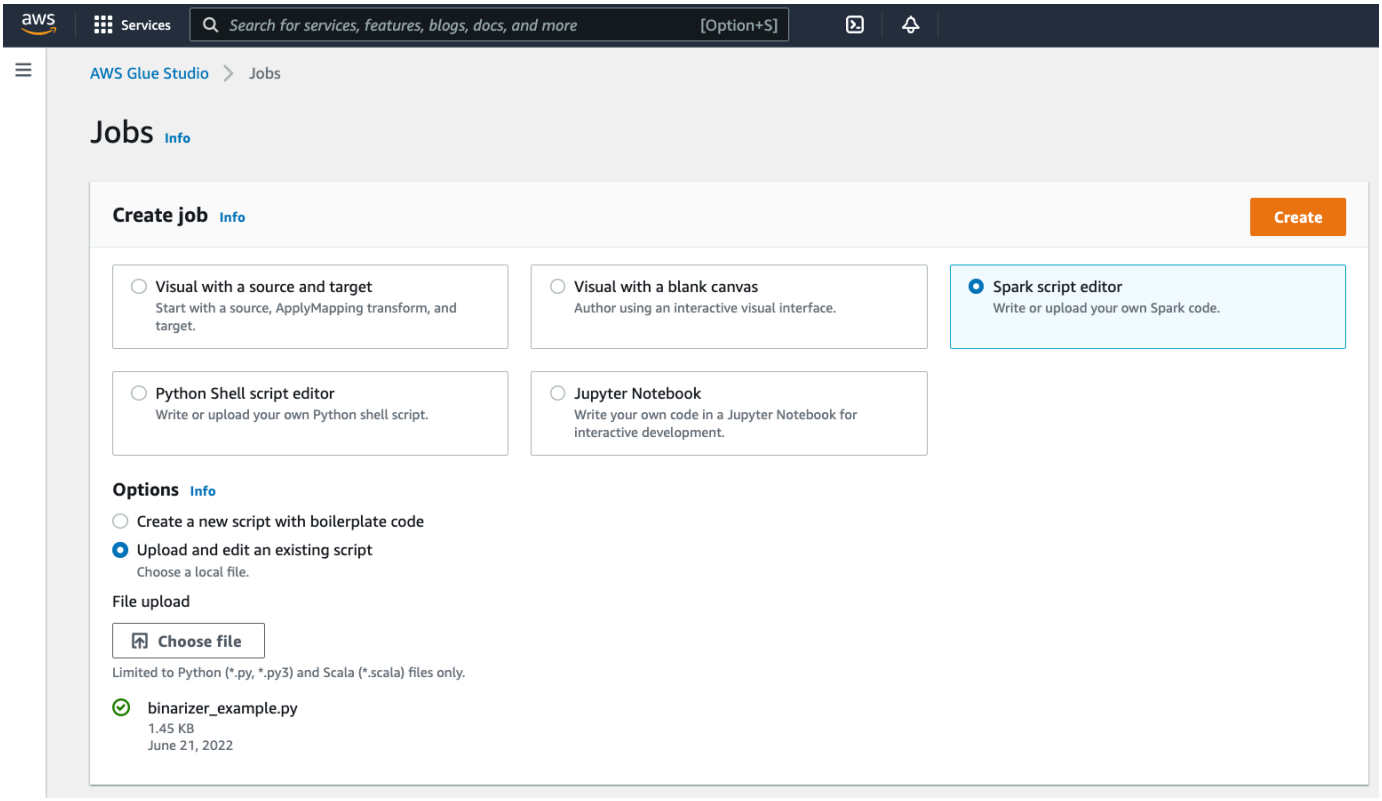
El código nativo Spark se puede ejecutar en un entorno de AWS Glue directamente. Los scripts a menudo se desarrollan al cambiar iterativamente un fragmento de código, un flujo de trabajo adecuado para una sesión interactiva. No obstante, el código existente es más adecuado para su ejecución en un trabajo de AWS Glue, que permite programar y obtener sistemáticamente registros y métricas para cada ejecución de script. Puede cargar y editar un script existente mediante la consola.

1. Adquiera el origen de su script. Para este ejemplo, utilizará un script de muestra del repositorio de Apache Spark. Ejemplo de binarizer
2. En la consola de AWS Glue, expanda el panel de navegación izquierdo y seleccione ETL > Jobs (Trabajos)

   En el panel Create job (Crear trabajo), seleccioneSpark script editor (Editor de scripts de Spark). Aparecerá la sección Options (Opciones). En Options (Opciones), elija Upload and edit an existing script (Cargar y editar un script existente).

   Aparecerá la sección File upload (Cargar archivo). En File upload (Cargar archivo), haga clic en Choose file (Elegir archivo). Aparecerá el selector de archivos del sistema. Vaya hasta la ubicación donde guardó `binarizer_example.py`, selecciónelo y confirme la opción elegida.

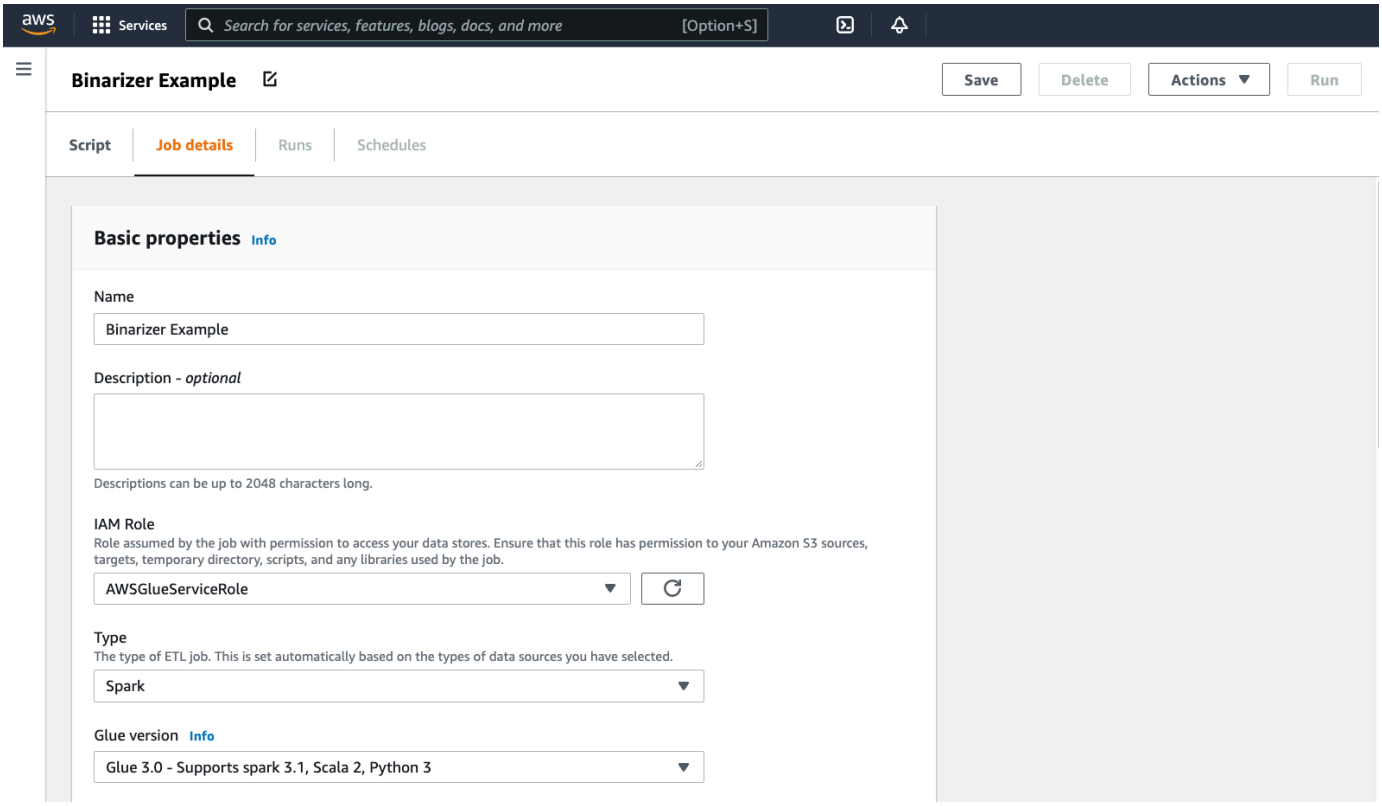
   Aparecerá el botón Create (Crear) en el encabezado del panel Create job (Crear trabajo). Haga clic allí.



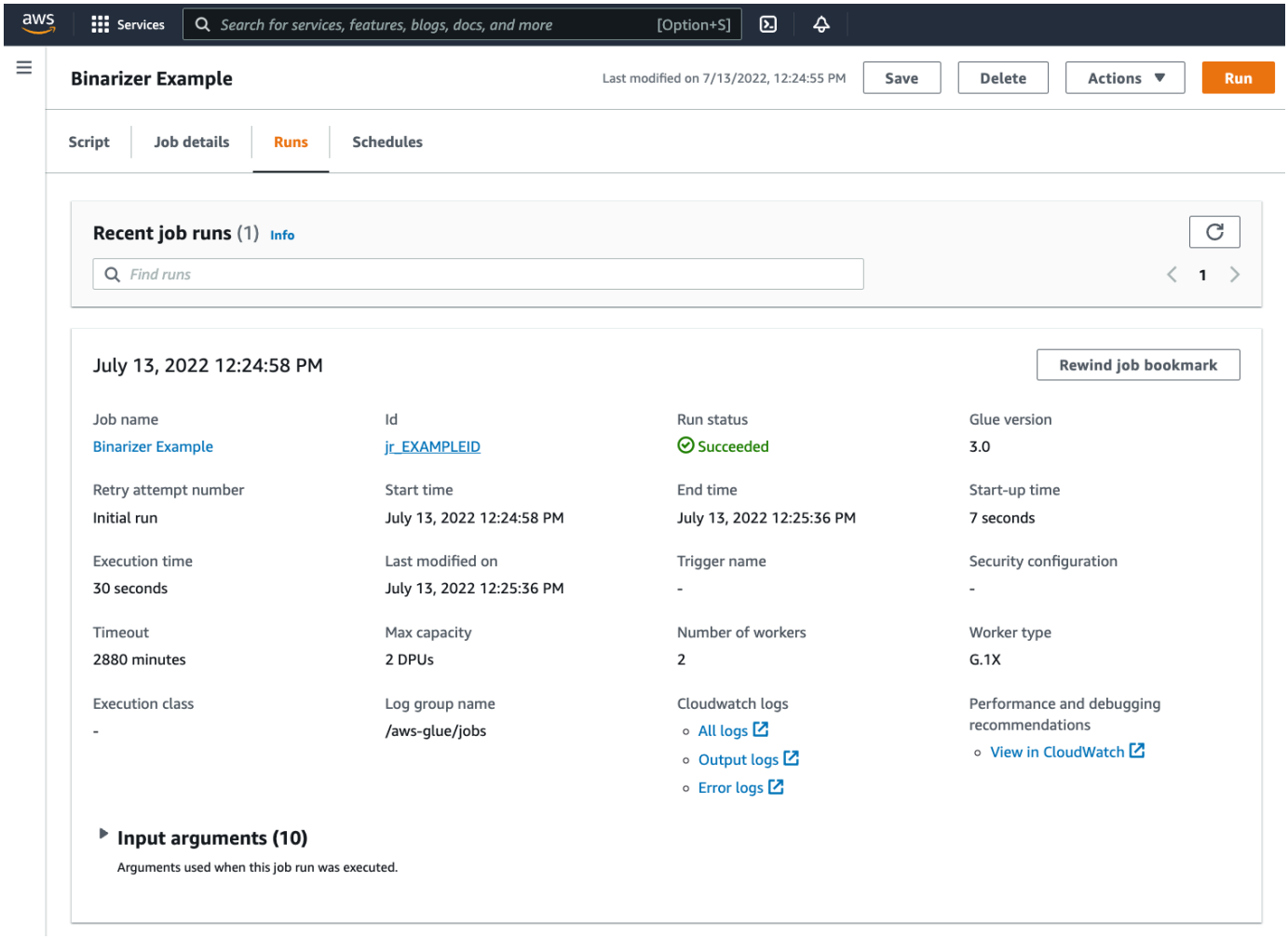
3. El navegador irá hasta el editor de scripts. En el encabezado, haga clic en la pestaña Job details (Detalles del trabajo). Defina el nombre y el Rol de IAM. Para obtener información sobre los roles de IAM de AWS Glue, consulte the section called “Configuración de permisos de IAM”.

   Opcionalmente: configure Requested number of workers (Cantidad solicitada de trabajadores) en 2 y Number of retries (Cantidad de reintentos) en 1. Estas opciones son valiosas a la hora de ejecutar trabajos de producción, pero rechazarlas agilizará la experiencia mientras prueba una función.

   En la barra de navegación, haga clic en Save (Guardar), luego en Run (Ejecutar)

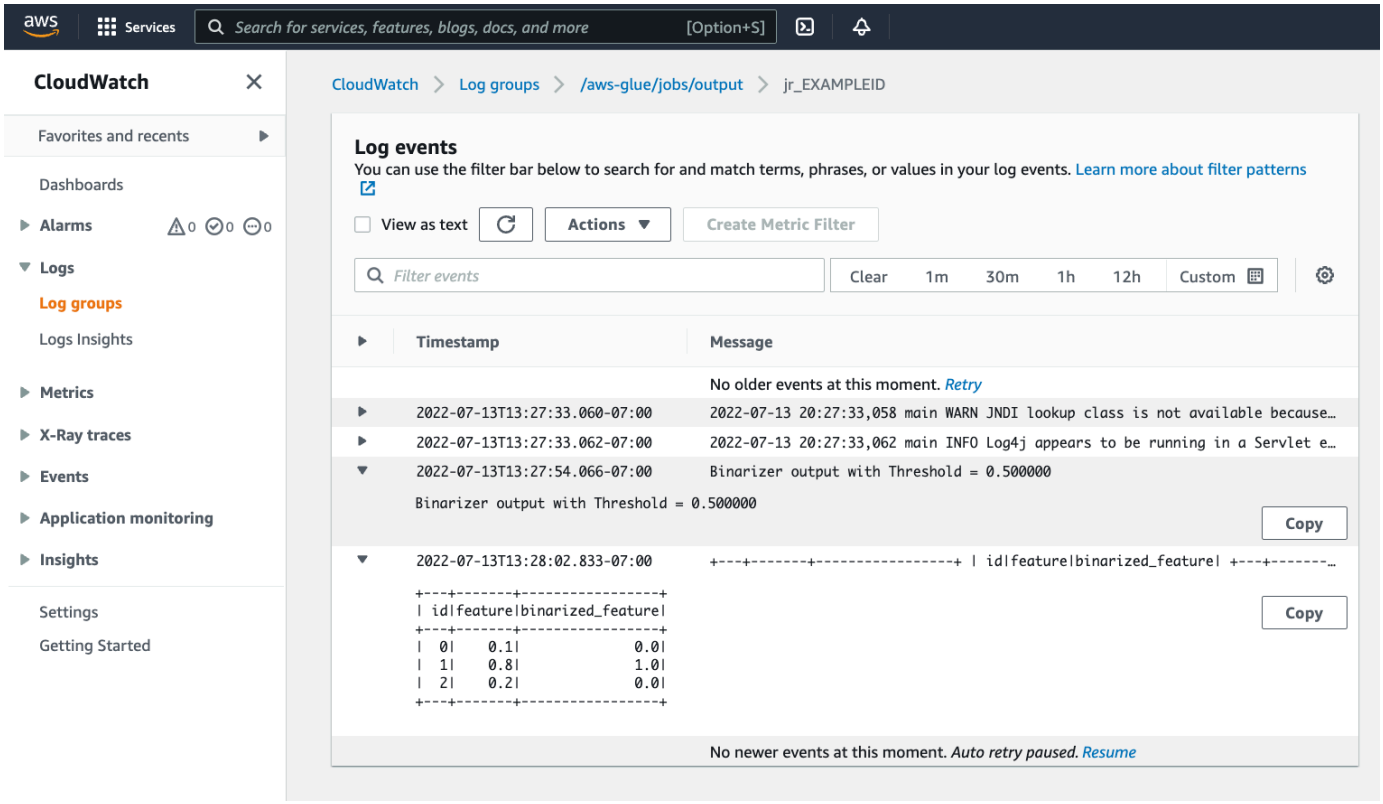


4. Vaya a la pestaña Runs (Ejecuciones). Verá un panel correspondiente a la ejecución de su trabajo. Espere unos minutos y la página deberá actualizarse automáticamente para mostrar Succeeded (Exitoso) en Run status (Estado de la ejecución).



5. Deberá examinar la salida para confirmar que el script de Spark se ejecutó según lo previsto. Este script de ejemplo de Apache Spark debe escribir una cadena en la secuencia de salida. Puede encontrarlo yendo a Output logs (Registros de salida) en Registros de Cloudwatch (Cloudwatch logs) en el panel para la ejecución correcta del trabajo. Tenga en cuenta el id de ejecución de trabajo, un id generado en la etiqueta Id (Identificador) que comienza con `jr_`.

   Esto abrirá la consola de CloudWatch, configurada para visualizar el contenido del AWS Gluegrupo de registro`/aws-glue/jobs/output` predeterminado, filtrado al contenido de los flujos de registro para el identificador de ejecución del trabajo. Cada trabajador habrá generado un flujo de registro, que se muestra como filas en Log streams (Flujos de registro). Un trabajador debería haber ejecutado el código solicitado. Deberá abrir todos los flujos de registro para identificar al trabajador correcto. Una vez que encuentre el trabajador correcto, se debería ver la salida del script, como se ve en la siguiente imagen:



Procedimientos comunes necesarios para migrar programas Spark

Evaluar el soporte de la versión de Spark

Las versiones de AWS Glue determinan la versión de Apache Spark y Python disponible para el trabajo de AWS Glue. Puede encontrar nuestras versiones de AWS Glue y lo que admiten en the section called “Versiones de AWS Glue”. Es posible que tenga que actualizar el programa de Spark para que sea compatible con una versión más reciente de Spark con el fin de acceder a determinadas características de AWS Glue.

Incluir bibliotecas de terceros

Muchos programas de Spark existentes tendrán dependencias, tanto en artefactos privados como públicos. AWS Glue admite dependencias de estilo JAR para trabajos de Scala, así como dependencias Wheel y de origen puro de Python para trabajos de Python.

Python: para obtener información sobre las dependencias de Python, consulte the section called “Bibliotecas Python”

Las dependencias habituales de Python se proporcionan en el entorno de AWS Glue, incluida la biblioteca Pandas que se suele solicitar. Estas dependencias se incluyen en AWS Glue versión

2.0+. Para obtener más información sobre los módulos proporcionados, consulte the section called "Módulos de Python ya proporcionados en AWS Glue". Si necesita suministrar un trabajo con una versión diferente de una dependencia incluida de manera predeterminada, puede utilizar `--additional-python-modules`. Para obtener información sobre argumentos de trabajo, consulte the section called "Parámetros del flujo de trabajo".

Puede proporcionar dependencias de Python adicionales con el argumento de trabajo `--extra-py-files`. Si va a migrar un trabajo desde un programa de Spark, este parámetro es una buena opción porque equivale funcionalmente al indicador `--py-files` de PySpark y está sujeto a las mismas limitaciones. Para obtener más información sobre el parámetro `--extra-py-files`, consulte the section called "Incluir archivos de Python con funciones PySpark nativas"

Para los nuevos trabajos, puede administrar las dependencias de Python con el argumento de trabajo `--additional-python-modules`. El uso de este argumento permite una experiencia de administración de dependencias más completa. Este parámetro admite dependencias de estilo Wheel, incluidas aquellas con enlaces de código nativo compatibles con Amazon Linux 2.

Scala

Puede proporcionar dependencias de Scala adicionales con el argumento de trabajo `--extra-jars`. Las dependencias deben estar alojadas en Amazon S3 y el valor del argumento debe ser una lista delimitada por comas de rutas de Amazon S3 sin espacios. Puede que le resulte más fácil administrar la configuración si reagrupa las dependencias antes de alojarlas y configurarlas. AWS Glue Las dependencias JAR contienen código de bytes de Java, que se puede generar desde cualquier lenguaje JVM. Puede usar otros lenguajes de JVM, como Java, para escribir dependencias personalizadas.

Administrar credenciales de origen de datos.

Los programas Spark existentes pueden incluir una configuración compleja o personalizada para extraer datos de sus fuentes de datos. Los flujos de autenticación de orígenes de datos habituales son compatibles con conexiones de AWS Glue. Para obtener más información acerca de las conexiones de AWS Glue, consulte Conexión a datos.

Las conexiones de AWS Glue facilitan la conexión de un trabajo a diversos tipos de almacenes de datos principalmente de dos formas: mediante llamadas de métodos a nuestras bibliotecas y configurando Additional network connection (Conexión de red adicional) en la consola de AWS. También puede llamar al AWS SDK desde su trabajo para recuperar información de una conexión.

Llamada de métodos: las conexiones de AWS Glue están estrechamente integradas con el Catálogo de datos de AWS Glue, un servicio que permite seleccionar información sobre los conjuntos de datos, y los métodos disponibles para interactuar con las conexiones de AWS Glue así lo reflejan. Si tiene una configuración de autenticación existente que le gustaría reutilizar, para conexiones JDBC, puede acceder a la configuración de conexión de AWS Glue mediante el método `extract_jdbc_conf` en `GlueContext`. Para obtener más información, consulte the section called "extract_jdbc_conf".

Configuración de la consola: los trabajos de AWS Glue utilizan las conexiones de AWS Glue asociadas para configurar conexiones a subredes de Amazon VPC. Si administra directamente sus materiales de seguridad, es posible que tenga que proporcionar una Additional network connection (Conexión de red adicional) de tipo `NETWORK` en la consola de AWS para configurar el enrutamiento. Para obtener más información sobre la API de conexión de AWS Glue, consulte the section called "Conexiones".

Si sus programas Spark tienen un flujo de autenticación personalizado o poco común, es posible que tenga que gestionar los materiales de seguridad de forma manual. Si las conexiones de AWS Glue no resultan adecuadas, se puede alojar de manera segura los materiales de seguridad en Secrets Manager y acceder a ellos a través de boto3 o el AWS SDK, que se proporcionan en el trabajo.

Configurar Apache Spark

Las migraciones complejas a menudo alteran la configuración de Spark para adaptarse a sus cargas de trabajo. Las versiones modernas de Apache Spark permiten configurar el tiempo de ejecución con `SparkSession`. AWS Glue Se proporciona un `SparkSession` a los trabajos 3.0+, que se puede modificar para establecer la configuración del tiempo de ejecución. Configuración Apache Spark. Ajustar Spark es complejo, y AWS Glue no garantiza asistencia para realizar toda la configuración de Spark. Si la migración requiere una configuración importante en el nivel de Spark, contacte con el servicio de asistencia.

Establecer configuración personalizada

Los programas de Spark migrados se pueden diseñar para que adopten una configuración personalizada. AWS Glue permite establecer la configuración en el nivel del trabajo y la ejecución del trabajo, mediante los argumentos del trabajo. Para obtener información sobre argumentos de trabajo, consulte the section called "Parámetros del flujo de trabajo". Puede acceder a los argumentos del trabajo dentro del contexto de un trabajo a través de nuestras bibliotecas. AWS Glue proporciona una función de utilidad para ofrecer una vista coherente de los argumentos establecidos en el trabajo y

aquellos establecidos en la ejecución del trabajo. Consulte the section called “getResolvedOptions” en Python y the section called “GlueArgParser” en Scala.

Migración de código Java

Como se explica en the section called “Bibliotecas de terceros”, sus dependencias pueden contener clases generadas por lenguajes JVM, como Java o Scala. Sus dependencias pueden incluir un método de `main`. Puede usar un método de `main` en una dependencia como punto de entrada para un trabajo de Scala de AWS Glue. Esto permite escribir su método de `main` en Java, o reutilizar un método de `main` empaquetado según los estándares de su propia biblioteca.

Para usar un método de `main` de una dependencia, realice lo siguiente: borre el contenido del panel de edición proporcionando el objeto predeterminado `GlueApp`. Proporcione el nombre completo de una clase en una dependencia como argumento de trabajo con la clave de `--class`. Luego, podrá activar una ejecución de trabajo.

No se puede configurar el orden ni la estructura de los argumentos que AWS Glue pasa al método `main`. Si el código existente necesita leer la configuración establecida en AWS Glue, es probable que esto provoque incompatibilidad con el código anterior. Si usa `getResolvedOptions`, tampoco tendrá un buen lugar para llamar a este método. Considere la posibilidad de invocar la dependencia directamente desde un método main generado por AWS Glue. El siguiente script de ETL de AWS Glue muestra un ejemplo de esto.

```
import com.amazonaws.services.glue.util.GlueArgParser

object GlueApp {
  def main(sysArgs: Array[String]) {
    val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB_NAME").toArray)

    // Invoke static method from JAR. Pass some sample arguments as a String[], one
 defined inline and one taken from the job arguments, using getResolvedOptions
    com.mycompany.myproject.MyClass.myStaticPublicMethod(Array("string parameter1",
 args("JOB_NAME")))

    // Alternatively, invoke a non-static public method.
    (new com.mycompany.myproject.MyClass).someMethod()
  }
}
```

Trabajar con tareas de Ray en AWS Glue

En esta sección se proporciona información sobre el uso de AWS Glue para trabajos de Ray. Para obtener más información sobre cómo escribir scripts de AWS Glue para Ray, consulte la sección the section called “AWS Glue para Ray”.

Temas

- Introducción a AWS Glue para Ray
- Entornos de tiempo de ejecución de Ray compatibles
- Contabilidad de los trabajadores en los trabajos de Ray
- Uso de los parámetros de trabajo en los trabajos de Ray
- Supervisión de trabajos de Ray con métricas

Introducción a AWS Glue para Ray

Para trabajar con AWS Glue para Ray, se utilizan los mismos trabajos de AWS Glue y las mismas sesiones interactivas que se usan con AWS Glue para Spark. Los trabajos de AWS Glue están diseñados para ejecutar el mismo script de forma periódica, mientras que las sesiones interactivas están diseñadas para permitir ejecutar fragmentos de código de forma secuencial con los mismos recursos aprovisionados.

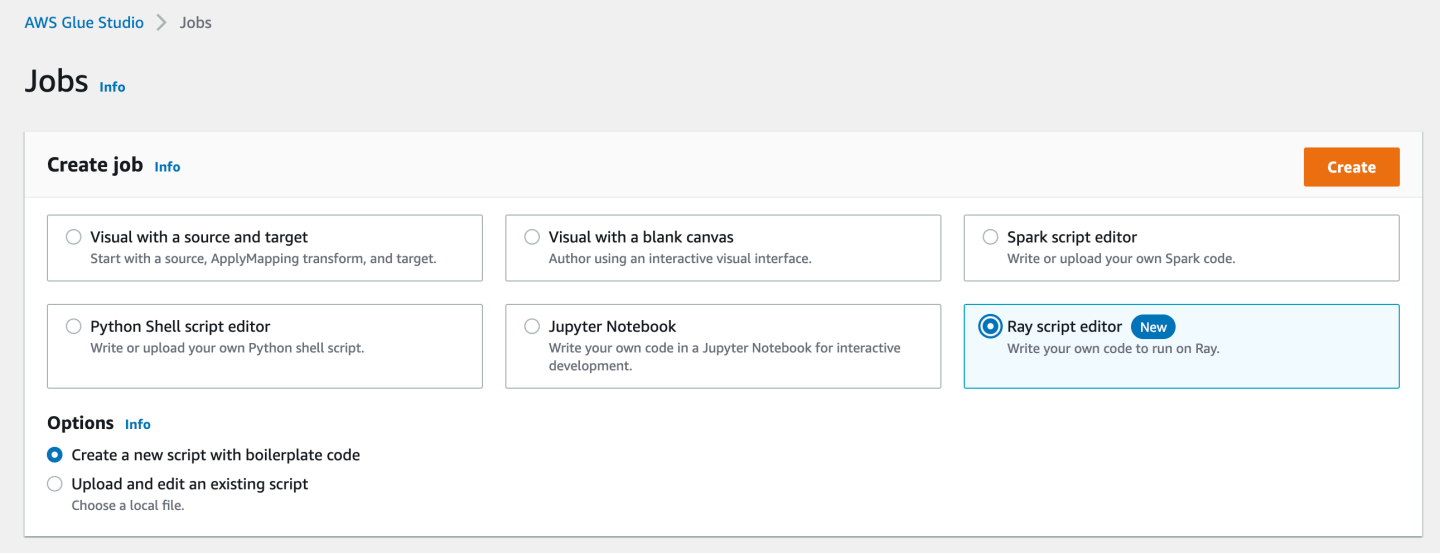
AWS Glue ETL y Ray son diferentes en el fondo, por lo que en el script tendrá acceso a diferentes herramientas, características y configuraciones. Como nuevo marco de cálculo administrado por AWS Glue, Ray tiene una arquitectura diferente y usa un vocabulario diferente para describir lo que hace. Para más información, consulte Architecture Whitepapers (Documentos técnicos sobre la arquitectura) en la documentación sobre Ray.

> **Note**
>
> AWS Glue para Ray está disponible en el Este de EE. UU. (Norte de Virginia), Este de EE. UU. (Ohio), Oeste de EE. UU. (Oregón), Asia Pacífico (Tokio) y Europa (Irlanda).

Trabajos de Ray en la consola de AWS Glue Studio

En la página Trabajos de la consola de AWS Glue Studio, puede seleccionar una nueva opción al crear un trabajo en el editor de scripts AWS Glue Studio Ray. Seleccione esta opción para crear un

trabajo de Ray en la consola. Para más información sobre los trabajos y cómo se utilizan, consulte Creación de trabajos de ETL visuales con AWS Glue Studio.



Trabajos de Ray en la AWS CLI y SDK

Los trabajos de Ray en la AWS CLI utilizan las mismas acciones y parámetros del SDK que los demás trabajos. AWS Glue para Ray introduce nuevos valores para determinados parámetros. Para más información sobre la API de trabajos, consulte the section called “Trabajos”.

Entornos de tiempo de ejecución de Ray compatibles

En los trabajos de Spark, `GlueVersion` determina las versiones de Apache Spark y Python disponibles en un trabajo de AWS Glue para Spark. La versión de Python indica la versión admitida para trabajos de tipo Spark. No es así como se configuran los entornos de tiempo de ejecución de Ray.

Para los trabajos de Ray, debe configurar `GlueVersion` en `4.0` o superior. Sin embargo, las versiones de Ray, Python y bibliotecas adicionales que están disponibles en el trabajo de Ray vienen determinadas por el campo `Runtime` de la definición del trabajo.

El entorno de tiempo de ejecución de `Ray2.4` estará disponible durante un mínimo de 6 meses después del lanzamiento. A medida que Ray evolucione rápidamente, podrá incorporar actualizaciones y mejoras de Ray en futuras versiones del entorno de tiempo de ejecución.

Valores válidos: `Ray2.4`

Valor de tiempo de ejecución	Versiones de Ray y Python
`Ray2.4` (para AWS Glue 4.0+)	Ray 2.4.0 Python 3.9

Información adicional

- Para ver las notas de la versión que acompañan a las versiones de AWS Glue para Ray, consulte the section called “Versiones de AWS Glue”.
- Para ver las bibliotecas de Python que se proporcionan en un entorno de tiempo de ejecución, consulte the section called “Módulos incluidos con los trabajos de Ray”.

Contabilidad de los trabajadores en los trabajos de Ray

AWS Glue ejecuta los trabajos de Ray en los nuevos tipos de trabajadores de EC2 basados en Graviton, que solo están disponibles para los trabajos de Ray. Para aprovisionar adecuadamente a estos trabajadores para las cargas de trabajo para las que Ray está diseñado, ofrecemos una proporción diferente entre los recursos de cómputo y los recursos de memoria que la mayoría de los trabajadores. Para tener en cuenta estos recursos, utilizamos la unidad de procesamiento de datos optimizada para la memoria (M-DPU) en lugar de la unidad de procesamiento de datos (DPU) estándar.

- Una M-DPU corresponde a 4 vCPU y 32 GB de memoria.
- Una DPU corresponde a 4 vCPU y 16 GB de memoria. Las DPU se utilizan para contabilizar los recursos en AWS Glue con los trabajos de Spark y los trabajadores correspondientes.

Actualmente, los trabajos de Ray tienen acceso a un tipo de trabajador, `Z.2X`. El trabajador `Z.2X` se asigna a 2 M-DPU (8 vCPU, 64 GB de memoria) y tiene 128 GB de espacio en disco. Una máquina `Z.2X` proporciona 8 trabajadores de Ray (uno por vCPU).

La cantidad de M-DPU que puede utilizar simultáneamente en una cuenta está sujeta a una cuota de servicio. Para más información acerca de los límites de su cuenta de AWS Glue, consulte Puntos de conexión y cuotas de AWS Glue.

La cantidad de nodos de trabajo que están disponibles para un trabajo de Ray con `--number-of-workers (NumberOfWorkers)` se especifica en la definición de trabajo. Para obtener más información acerca de los valores Ray en la API de trabajos, consulte the section called “Trabajos”.

También puede especificar un número mínimo de trabajadores que un trabajo de Ray debe asignar con el parámetro de trabajo de `--min-workers`. Para obtener más información acerca de la configuración de parámetros de trabajos, consulte the section called “Referencia”.

Uso de los parámetros de trabajo en los trabajos de Ray

Establece los argumentos para los trabajos de Ray AWS Glue de la misma manera que establece los argumentos para AWS Glue para los trabajos de Spark. Para obtener más información sobre la API de AWS Glue, consulte the section called “Trabajos”. Puede configurar los trabajos de Ray AWS Glue con diferentes argumentos, que se enumeran en esta referencia. También puede presentar sus propios argumentos.

Puede configurar un trabajo en la consola, en la pestaña Job details (Detalles del trabajo), en el encabezado Job Parameters (Parámetros del trabajo). También puede configurar un trabajo mediante la AWS CLI al establecer `DefaultArguments` en un trabajo o `Arguments` en una ejecución de trabajos. Los parámetros de los trabajos y los argumentos predeterminados permanecerán con el trabajo durante varias ejecuciones.

Por ejemplo, a continuación se muestra la sintaxis para ejecutar un trabajo con `--arguments` para configurar un parámetro especial.

```
$ aws glue start-job-run --job-name "CSV to CSV" --arguments='--scriptLocation="s3://
my_glue/libraries/test_lib.py",--test-environment="true"'
```

Después de establecer los argumentos, puede acceder a los parámetros del trabajo desde su trabajo de Ray mediante variables de entorno. Esto permite configurar el trabajo para cada ejecución. El nombre de la variable de entorno será el nombre del argumento del trabajo sin el prefijo `--`.

Por ejemplo, en el anterior, los nombres de las variables serían `scriptLocation` y `test-environment`. Luego, recuperaría el argumento mediante los métodos disponibles en la biblioteca estándar: `test_environment = os.environ.get('test-environment')`. Para más información sobre el acceso a las variables de entorno con Python, consulte módulo del sistema operativo en la documentación de Python.

Configure la forma en que los trabajos de Ray generan registros

De forma predeterminada, los trabajos de Ray generan registros y métricas que se envían a CloudWatch y Amazon S3. Puede usar el parámetro `--logging_configuration` para modificar la forma en que se generan los registros; actualmente, puede usarlo para evitar que los trabajos de Ray generen varios tipos de registros. Este parámetro toma un objeto JSON, cuyas claves corresponden a los registros o comportamientos que desea modificar. Admite las siguientes claves:

- `CLOUDWATCH_METRICS`: configura las series de métricas de CloudWatch que se pueden usar para visualizar el estado del trabajo. Para obtener más información sobre las métricas, consulte the section called “Métricas de trabajo de Ray”.
- `CLOUDWATCH_LOGS`: configura los registros de CloudWatch que proporcionan detalles a nivel de aplicación de Ray acerca del estado de ejecución del trabajo. Para obtener más información acerca de los registros, consulte the section called “Solución de errores de Ray”.
- `S3`: configura lo que AWS Glue escribe en Amazon S3, principalmente información similar a la de los registros de CloudWatch, pero como archivos y no como flujos de registros.

Para deshabilitar un comportamiento de registro de Ray, indique el valor `{\"IS_ENABLED\": \"False\"}`. Por ejemplo, para deshabilitar las métricas y los registros de CloudWatch, proporcione la siguiente configuración:

```
"--logging_configuration": "{\"CLOUDWATCH_METRICS\": {\"IS_ENABLED\": \"False\"},
 \"CLOUDWATCH_LOGS\": {\"IS_ENABLED\": \"False\"}}"
```

Referencia

Los trabajos de Ray reconocen los siguientes nombres de argumentos que se pueden usar para configurar el entorno de scripts para los trabajos de Ray y las ejecuciones de trabajos:

- `--logging_configuration`: se usa para detener la generación de varios registros creados por trabajos de Ray. Estos registros se generan de forma predeterminada en todos los trabajos de Ray. Formato: objeto JSON incluido en una cadena de escape. Para obtener más información, consulte the section called “Configure la forma en que los trabajos de Ray generan registros”.
- `--min-workers`: la cantidad mínima de nodos de trabajo que se asignan a un trabajo de Ray. Un nodo de trabajo puede ejecutar varias réplicas, una por CPU virtual. Formato: entero. Mínimo: 0. Máximo: valor especificado en `--number-of-workers (NumberOfWorkers)` en la definición

del trabajo. Para obtener más información sobre la contabilidad de los nodos de trabajo, consulte the section called “Contabilidad de los trabajadores en los trabajos de Ray”.

- `--object_spilling_config`: AWS Glue porque Ray admite el uso de Amazon S3 como una forma de ampliar el espacio disponible para el almacén de objetos de Ray. Para habilitar este comportamiento, puede proporcionar a Ray un objeto que vuelque un objeto de configuración de JSON con este parámetro. Para obtener más información sobre la configuración de volcado de objetos de Ray, consulte Derrame de objetos en la documentación de Ray. Formato: objeto JSON.

  AWS Glue para Ray solo admite el volcado en el disco o en Amazon S3 a la vez. Puede proporcionar varios lugares para el volcado, siempre y cuando respeten esta limitación. Cuando vuelque a Amazon S3, también necesitará agregar permisos de IAM al trabajo para este bucket.

  Al proporcionar un objeto JSON como configuración con la CLI, debe proporcionarlo como una cadena, con el objeto JSON incluido en una cadena de escape. Por ejemplo, un valor de cadena para volcar en una ruta de Amazon S3 tendría el siguiente aspecto: `"{\"type\": \"smart_open\", \"params\": {\"uri\":\"s3path\"}}"`. En AWS Glue Studio, proporcione este parámetro como un objeto JSON sin formato adicional.
- `--object_store_memory_head`: la memoria asignada al almacén de objetos de Plasma en el nodo principal de Ray. Esta instancia ejecuta servicios de administración de clústeres, así como réplicas de procesos de trabajo. El valor representa un porcentaje de memoria libre en la instancia después de un inicio en caliente. Este parámetro se utiliza para ajustar las cargas de trabajo de uso intensivo de la memoria. Los valores predeterminados son aceptables para la mayoría de los casos de uso. Formato: entero positivo. Mínimo: 1. Máximo: 100.

  Para más información sobre Plasma, consulte The Plasma In-Memory Object Store (Almacén de objetos en memoria de Plasma) en la documentación de Ray.
- `--object_store_memory_worker`: memoria asignada al almacén de objetos de Plasma en los nodos de trabajo de Ray. Estas instancias solo ejecutan réplicas de procesos de trabajo. El valor representa un porcentaje de memoria libre en la instancia después de un inicio en caliente. Este parámetro se utiliza para ajustar las cargas de trabajo de uso intensivo de la memoria. Los valores predeterminados son aceptables para la mayoría de los casos de uso. Formato: entero positivo. Mínimo: 1. Máximo: 100.

  Para más información sobre Plasma, consulte The Plasma In-Memory Object Store (Almacén de objetos en memoria de Plasma) en la documentación de Ray.
- `--pip-install`: un conjunto de paquetes de Python para instalar. Puede instalar paquetes desde PyPI con este argumento. Formato: lista delimitada por comas.

Una entrada de paquete de PyPI tiene el formato `package==version`, con el nombre y la versión de PyPI del paquete de destino. Las entradas utilizan la coincidencia de versiones de Python para hacer coincidir el paquete y la versión, como ==, no la igualdad simple =. Hay otros operadores de coincidencia de versión. Para más información, consulte PEP 440 en el sitio web de Python. También puede proporcionar módulos personalizados con `--s3-py-modules`.

- `--s3-py-modules`: un conjunto de rutas de Amazon S3 que alojan distribuciones de módulos de Python. Formato: lista delimitada por comas.

  Puede usar esto para distribuir sus propios módulos en el trabajo de Ray. También puede proporcionar módulos desde PyPI con `--pip-install`. A diferencia de ETL de AWS Glue, los módulos personalizados no se configuran mediante pip, sino que se pasan a Ray para ser distribuidos. Para obtener más información, consulte the section called “Módulos de Python adicionales para trabajos de Ray”.

- `--working-dir`: una ruta a un archivo .zip alojado en Amazon S3 que contiene archivos que se van a distribuir a todos los nodos que ejecutan el trabajo de Ray. Formato: cadena. Para obtener más información, consulte the section called “Proporcionar archivos a su trabajo de Ray”.

Supervisión de trabajos de Ray con métricas

Puede supervisar los trabajos de Ray mediante AWS Glue Studio y Amazon CloudWatch. CloudWatch recopila y procesa las métricas sin formato desde AWS Glue con Ray, lo que las hace disponibles para el análisis. Estas métricas se visualizan en la consola de AWS Glue Studio, por lo que puede supervisar el trabajo a medida que se ejecuta.

Para obtener información general sobre cómo supervisar AWS Glue, consulte the section called “Uso de métricas de CloudWatch”. Para obtener una descripción general de cómo utilizar las métricas de CloudWatch publicadas por AWS Glue, consulte the section called “Supervisión con CloudWatch”.

Supervisión de los trabajos de Ray en la consola de AWS Glue

En la página de detalles para una ejecución de trabajo, debajo de la sección Detalles de ejecución, puede consultar los gráficos agregados prediseñados que visualizan las métricas de trabajo disponibles. AWS Glue Studio envía las métricas del trabajo a CloudWatch para cada ejecución de trabajo. Con ellas, puede crear un perfil del clúster y las tareas y acceder a información detallada sobre cada nodo.

Para obtener más información sobre los gráficos de métricas disponibles, consulte the section called "Visualización de métricas de Amazon CloudWatch para una ejecución de trabajo de Ray".

Información general de las métricas de trabajos de Ray en CloudWatch

Publicamos las métricas de Ray cuando la supervisión detallada está habilitada en CloudWatch. Las métricas se publican en el espacio de nombres de CloudWatch `Glue/Ray`.

- Métricas de la instancia

  Publicamos métricas acerca del uso de la CPU, la memoria y el disco de las instancias asignadas a un trabajo. Estas métricas se identifican mediante características como `ExecutorId`, `ExecutorType` y `host`. Estas métricas son un subconjunto de las métricas estándar de los agentes de CloudWatch de Linux. Puede encontrar información sobre los nombres y características de las métricas en la documentación de CloudWatch. Para más información, consulte Métricas que el agente de CloudWatch ha recopilado.

- Métricas del clúster de Ray

  Reenviamos las métricas de los procesos de Ray que ejecutan el script en este espacio de nombres, luego proporcionamos los más críticos. Las métricas disponibles pueden diferir según la versión de Ray. Para más información sobre qué versión de Ray está en ejecución en su trabajo, consulte the section called "Versiones de AWS Glue".

  Ray recopila métricas al nivel de instancia. También proporciona métricas para las tareas y el clúster. Para obtener más información sobre la estrategia de métricas subyacente de Ray, consulte Métricas en la documentación de Ray.

> **Note**
>
> No publicamos las métricas de Ray en el espacio de nombres de `Glue/Job Metrics/`, que solo se usa para trabajos de ETL de AWS Glue.

Configuración de las propiedades de trabajos del intérprete de comandos de Python en AWS Glue

Un trabajo de shell de Python puede usarse para ejecutar scripts de Python como un shell en AWS Glue. Con un trabajo del intérprete de comandos de Python puede ejecutar scripts compatibles con Python 3.6 o Python 3.9.

Temas

- Limitaciones
- Definición de las propiedades de trabajos de shell de Python
- Bibliotecas compatibles con trabajos de shell de Python
- Proporcionar su propia biblioteca de Python
- Use AWS CloudFormation con los trabajos del intérprete de comandos de Python en AWS Glue

Limitaciones

Tenga en cuenta las siguientes limitaciones de los trabajos de Python Shell:

- No se pueden utilizar marcadores de trabajos con los trabajos de shell de Python.
- No se puede empaquetar ninguna biblioteca de Python como archivos `.egg` en Python 3.9 y superiores. En su lugar, utilice `.whl`.
- No se puede usar la opción `--extra-files` debido a una limitación en las copias temporales de los datos de S3.

Definición de las propiedades de trabajos de shell de Python

Estas secciones describen la definición de propiedades de trabajo en AWS Glue Studio, o mediante el CLI de AWS.

AWS Glue Studio

Al definir su trabajo de shell de Python en AWS Glue Studio, debe proporcionar algunas de las siguientes propiedades:

Rol de IAM

Especifique el rol de AWS Identity and Access Management (IAM) que se usa para dar una autorización a los recursos que se utilizan para ejecutar el flujo de trabajo y obtener acceso a los almacenes de datos. Para obtener más información acerca de los permisos para ejecutar trabajos en AWS Glue, consulte Gestión de identidad y acceso para AWS Glue.

Tipo

Elija Python shell (Shell de Python) para ejecutar un script de Python con el comando de trabajo llamado `pythonshell`.

Versión de Python

Elija la versión de Python. El valor por defecto es Python 3.9. Las versiones válidas son Python 3.6 y Python 3.9.

Cargar bibliotecas de análisis comunes (recomendado)

Elija esta opción para incluir bibliotecas comunes para Python 3.9 en el shell de Python.

Si sus bibliotecas son personalizadas o entran en conflicto con las preinstaladas, puede optar por no instalar bibliotecas comunes. Sin embargo, puede instalar bibliotecas adicionales además de las bibliotecas comunes.

Si selecciona esta opción, la opción `library-set` se establece en `analytics`. Al anular la selección de esta opción, la opción `library-set` se establece en `none`.

Nombre de archivo de script y ruta de script

El código del script define la lógica de procedimiento del trabajo. Proporcione el nombre del script y la ubicación en Amazon Simple Storage Service (Amazon S3). Compruebe que no haya un archivo con el mismo nombre que el directorio de script en la ruta. Para obtener más información acerca de cómo usar scripts, consulte Guía de programación de AWS Glue.

Script

El código del script define la lógica de procedimiento del trabajo. Puede codificar el script en Python 3.6 o Python 3.9. Puede editar scripts en AWS Glue Studio.

Unidades de procesamiento de datos

El número máximo de unidades de procesamiento de datos (DPU) de AWS Glue que se pueden asignar cuando se ejecute este trabajo. Una DPU es una medida relativa de la potencia de

procesamiento que consta de 4 vCPU de capacidad de cómputo y 16 GB de memoria. Para más información, consulte Precios de AWS Glue.

Puede establecer el valor en 0.0625 o 1. El valor predeterminado es 0.0625. En cualquier caso, el disco local para la instancia será de 20 GB.

CLI

También puede crear un trabajo de shell de Python con la AWS CLI, como en el siguiente ejemplo.

```
aws glue create-job --name python-job-cli --role Glue_DefaultRole
    --command '{"Name" :  "pythonshell", "PythonVersion": "3.9", "ScriptLocation" :
"s3://DOC-EXAMPLE-BUCKET/scriptname.py"}'
    --max-capacity 0.0625
```

> **Note**
>
> No es necesario especificar la versión de AWS Glue ya que el parámetro `--glue-version` no se aplica a los trabajos del intérprete de comandos de AWS Glue. Se ignorará cualquier versión especificada.

Trabajos que crea con el valor AWS CLI predeterminado en Python 3. Las versiones válidas de Python son 3 (que corresponden a 3.6) y 3.9. Para especificar Python 3.6, agregue esta tupla al parámetro `--command`: `"PythonVersion":"3"`

Para especificar Python 3.9, agregue esta tupla al parámetro `--command`: `"PythonVersion":"3.9"`

Para establecer la capacidad máxima utilizada por un trabajo de shell de Python, proporcione el parámetro `--max-capacity`. En los trabajos de shell de Python, no se puede utilizar el parámetro `--allocated-capacity`.

Bibliotecas compatibles con trabajos de shell de Python

En el shell de Python con Python 3.9, puede elegir el conjunto de bibliotecas para usar los conjuntos de bibliotecas preempaquetados según sus necesidades. Puede utilizar la opción `library-set` para elegir el conjunto de bibliotecas. Los valores válidos son `analytics` y `none`.

El entorno para la ejecución de un trabajo de shell de Python admite las siguientes bibliotecas:

Versión de Python	Python 3.6	Python 3.9	
Conjunto de bibliotecas	N/D	análisis	ninguno
avro		1.11.0	
awscli	116.242	1.23.5	1.23.5
awswrangler		2.15.1	
botocore	1.12.232	1.24.21	1.23.5
boto3	1.9.203	1.21.21	
elasticsearch		8.2.0	
numpy	1.16.2	1.22.3	
pandas	0.24.2	1.4.2	
psycopg2		2.9.3	
pyathena		2.5.3	
PyGreSQL	5.0.6		
PyMySQL		1.0.2	
pyodbc		4.0.32	
pyorc		0.6.0	
redshift-connector		2.0.907	
solicitudes	2.22.0	2.27.1	
scikit-learn	0.20.3	1.0.2	
scipy	1.2.1	1.8.0	

Versión de Python	Python 3.6	Python 3.9	
SQLAlchemy		1.4.36	
s3fs		2022.3.0	

Puede utilizar la biblioteca NumPy en un trabajo de shell de Python para cálculos científicos. Para obtener más información, consulte NumPy. En el siguiente ejemplo se muestra un script NumPy que puede utilizarse en un trabajo de shell de Python. El script imprime "Hello world" y los resultados de varios cálculos matemáticos.

```
import numpy as np
print("Hello world")

a = np.array([20,30,40,50])
print(a)

b = np.arange( 4 )

print(b)

c = a-b

print(c)

d = b**2

print(d)
```

Proporcionar su propia biblioteca de Python

Uso de PIP

El shell de Python que usa Python 3.9 también le permite proporcionar módulos de Python adicionales o versiones diferentes al nivel de trabajo. Puede utilizar la opción `--additional-python-modules` con una lista de módulos Python separados por comas para agregar un nuevo módulo o cambiar la versión de un módulo existente. No puede proporcionar módulos de Python personalizados alojados en Amazon S3 con este parámetro al usar trabajos de intérprete de comandos de Python.

Por ejemplo, para actualizar o agregar un nuevo módulo `scikit-learn`, utilice la siguiente clave y valor: `"--additional-python-modules", "scikit-learn==0.21.3"`.

AWS Glue utiliza el instalador de paquetes de Python (pip3) para instalar los módulos adicionales. Puede incluir opciones de pip3 adicionales dentro del valor `--additional-python-modules`. Por ejemplo, `"scikit-learn==0.21.3 -i https://pypi.python.org/simple/"`. Se aplican incompatibilidades o limitaciones de pip3.

> **Note**
>
> Para evitar incompatibilidades en el futuro, recomendamos usar bibliotecas creadas para Python 3.9.

Uso de un archivo Egg o Whl

Es posible que ya tenga una o varias bibliotecas de Python empaquetadas como un archivo `.egg` o `.whl`. En caso afirmativo, puede especificarlos en su trabajo utilizando la AWS Command Line Interface (AWS CLI) bajo la marca `"--extra-py-files"`, como en el siguiente ejemplo.

```
aws glue create-job --name python-redshift-test-cli --role role --command '{"Name" :
 "pythonshell", "ScriptLocation" : "s3://MyBucket/python/library/redshift_test.py"}'
    --connections Connections=connection-name --default-arguments '{"--extra-py-
files" : ["s3://DOC-EXAMPLE-BUCKET/EGG-FILE", "s3://DOC-EXAMPLE-BUCKET/WHEEL-FILE"]}'
```

Si no está seguro de cómo crear un archivo `.egg` o `.whl` desde una biblioteca de Python, siga estos pasos. Este ejemplo es aplicable en macOS, Linux y Windows Subsystem for Linux (WSL).

Para crear un archivo .egg o.whl de Python

1. Cree un clúster de Amazon Redshift en una nube virtual privada (VPC) y agregue algunos datos a una tabla.
2. Cree una conexión de AWS Glue para la combinación VPC-GrupoDeSeguridad-Subred que utilizó para crear el clúster. Compruebe que la conexión se realiza correctamente.
3. Cree un directorio llamado `redshift_example` y cree un archivo denominado `setup.py`. Pegue el siguiente código en `setup.py`.

   ```
   from setuptools import setup
   ```

```
setup(
    name="redshift_module",
    version="0.1",
    packages=['redshift_module']
)
```

4. En el directorio `redshift_example`, cree un directorio `redshift_module`. En el directorio `redshift_module`, cree los archivos `__init__.py` y `pygresql_redshift_common.py`.
5. Deje vacío el archivo `__init__.py`. En `pygresql_redshift_common.py`, pegue el siguiente código: Sustituya *`port`*, *`db_name`*, *`user`*, y *`password_for_user`* con los detalles específicos de su clúster de Amazon Redshift. Sustituya *`table_name`* por el nombre de la tabla en Amazon Redshift.

```
import pg


def get_connection(host):
    rs_conn_string = "host=%s port=%s dbname=%s user=%s password=%s" % (
        host, port, db_name, user, password_for_user)

    rs_conn = pg.connect(dbname=rs_conn_string)
    rs_conn.query("set statement_timeout = 1200000")
    return rs_conn


def query(con):
    statement = "Select * from table_name;"
    res = con.query(statement)
    return res
```

6. Si no está ya en él, cambie al directorio `redshift_example`.
7. Realice una de las siguientes acciones siguientes:

   - Ejecute el siguiente comando para crear un archivo `.egg`.

     ```
     python setup.py bdist_egg
     ```

   - Ejecute el siguiente comando para crear un archivo `.whl`.

     ```
     python setup.py bdist_wheel
     ```

8. Instale las dependencias necesarias para el comando anterior.
9. El comando crea un archivo en el directorio `dist`:

   - Si ha creado un archivo egg, se llamará `redshift_module-0.1-py2.7.egg`.
   - Si ha creado un archivo wheel, se llamará `redshift_module-0.1-py2.7-none-any.whl`.

   Cargue este archivo en Amazon S3.

   En este ejemplo, la ruta del archivo cargado es *`s3://DOC-EXAMPLE-BUCKET/EGG-FILE`* o *`s3://DOC-EXAMPLE-BUCKET/WHEEL-FILE`*.
10. Cree un archivo de Python que se debe usar como script para el trabajo de AWS Glue y añada el siguiente código al archivo.

```
from redshift_module import pygresql_redshift_common as rs_common

con1 = rs_common.get_connection(redshift_endpoint)
res = rs_common.query(con1)

print "Rows in the table cities are: "

print res
```

11. Cargue el archivo anterior en Amazon S3. En este ejemplo, la ruta del archivo cargado es *`s3://DOC-EXAMPLE-BUCKET/scriptname.py`*.
12. Cree un trabajo de shell de Python con este script. En la consola de AWS Glue, en la página Job properties (Propiedades del trabajo), especifique la ruta del archivo `.egg/.whl` en el cuadro Python library path (Ruta de la biblioteca Python). Si tiene varios archivos `.egg/.whl` y archivos de Python, proporcione una lista separada por comas en este cuadro.

    Al modificar o cambiar el nombre de los archivos `.egg`, los nombres de archivo deben usar los nombres predeterminados generados por el comando "python setup.py bdist_egg" o deben cumplir con las convenciones de nomenclatura del módulo Python. Para obtener más información, consulte Style Guide for Python Code.

    Con la AWS CLI, cree un trabajo con un comando, como en el siguiente ejemplo.

```
aws glue create-job --name python-redshift-test-cli --role Role --command
 '{"Name" :  "pythonshell", "ScriptLocation" : "s3://DOC-EXAMPLE-BUCKET/
scriptname.py"}'
     --connections Connections="connection-name" --default-arguments '{"--extra-
py-files" : ["s3://DOC-EXAMPLE-BUCKET/EGG-FILE", "s3://DOC-EXAMPLE-BUCKET/WHEEL-
FILE"]}'
```

Cuando se ejecuta el trabajo, el script imprime las filas creadas en la tabla *table_name* en el clúster de Amazon Redshift.

Use AWS CloudFormation con los trabajos del intérprete de comandos de Python en AWS Glue

Puede usar AWS CloudFormation con los trabajos del intérprete de comandos de Python en AWS Glue. A continuación, se muestra un ejemplo:

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  Python39Job:
    Type: 'AWS::Glue::Job'
    Properties:
      Command:
        Name: pythonshell
        PythonVersion: '3.9'
        ScriptLocation: 's3://bucket/location'
      MaxRetries: 0
      Name: python-39-job
      Role: RoleName
```

El grupo de Amazon CloudWatch Logs para la salida de los trabajos de shell de Python es `/aws-glue/python-jobs/output`. En el caso de los errores, consulte el grupo de registros `/aws-glue/python-jobs/error`.

Supervisión de AWS Glue

El monitoreo es una parte importante del mantenimiento de la fiabilidad, la disponibilidad y el desempeño de AWS Glue y otras soluciones de AWS. AWS ofrece herramientas de monitorización

que puede utilizar para controlar a AWS Glue, informar cuando algo no funciona y realizar acciones automáticamente cuando proceda:

Puede utilizar las siguientes herramientas de monitorización automatizado para vigilar AWS Glue e informar cuando haya algún problema:

- Amazon CloudWatch Events proporciona un flujo de eventos de sistema casi en tiempo real que describe cambios en los recursos de AWS. CloudWatch Events habilita una informática basada en eventos automatizada. Puede escribir reglas que vigilan determinados eventos y activan acciones automatizadas en otros servicios de AWS cuando estos eventos se producen. Para obtener más información, consulte la Guía del usuario de los Eventos de Amazon CloudWatch.
- Amazon CloudWatch Logs le permite monitorear, almacenar y tener acceso a los archivos de registro desde instancias de Amazon EC2, AWS CloudTrail u otros orígenes. Los registros de CloudWatch pueden monitorear información en los registros y enviarle una notificación cuando se llega a determinados umbrales. También se pueden archivar los datos del registro en un almacenamiento de larga duración. Para obtener más información, consulte la Guía del usuario de Registros de Amazon CloudWatch.
- AWS CloudTrail captura llamadas a la API y eventos relacionados efectuados por su cuenta de AWS o en su nombre, y entrega los archivos de registro al bucket de Amazon S3 que se haya especificado. También se puede identificar qué usuarios y cuentas llaman a AWS, la dirección IP de origen desde la que se realizan las llamadas y el momento en que se efectúan las llamadas. Para más información, consulte la Guía del usuario de AWS CloudTrail.

Además, tiene acceso a la siguiente información de la consola de AWS Glue que le ayudará a depurar y perfilar los trabajos:

- Trabajos de Spark: puede visualizar series de métricas seleccionadas de CloudWatch y los trabajos más nuevos tienen acceso a la interfaz de usuario de Spark. Para obtener más información, consulte the section called “Monitoreo de trabajos de Spark”.
- Trabajos de Ray: puede visualizar las series de métricas seleccionadas de CloudWatch. Para obtener más información, consulte the section called “Métricas de trabajo de Ray”.

Temas

- Etiquetas de AWS en AWS Glue
- Automatización de AWS Glue con CloudWatch Events
- Monitoreo de recursos AWS Glue

- Registro de llamadas a la API de AWS Glue con AWS CloudTrail

Etiquetas de AWS en AWS Glue

Para que le resulte más fácil administrar los recursos de AWS Glue, también puede asignar sus propias etiquetas a algunos tipos de recursos de AWS Glue. Una etiqueta es una marca que se asigna a un recurso de AWS. Cada etiqueta está formada por una clave y un valor opcional, ambos definidos por el usuario. Puede utilizar etiquetas de AWS Glue para organizar e identificar los recursos. Las etiquetas se pueden utilizar para crear informes de contabilidad de costos y restringir el acceso a los recursos. Si utiliza AWS Identity and Access Management, puede controlar qué usuarios de su cuenta de AWS tienen permiso para crear, editar o eliminar etiquetas. Además de los permisos para llamar a las API relacionadas con etiquetas, también necesita el permiso `glue:GetConnection` para llamar a las API de etiquetado en las conexiones y el permiso `glue:GetDatabase` para llamar a las API de etiquetado en las bases de datos. Para obtener más información, consulte ABAC con Glue AWS.

En AWS Glue, se pueden etiquetar los siguientes recursos:

- Connection
- Base de datos
- Rastreador
- Sesión interactiva
- Punto de enlace de desarrollo
- Trabajo
- Desencadenador
- Flujo de trabajo
- Esquema
- Transformación de machine learning
- Conjunto de reglas de calidad de datos
- Esquemas de streaming
- Registros de esquemas de streaming

> **Note**
>
> Como práctica recomendada, para permitir el etiquetado de estos recursos de AWS Glue, incluya siempre la acción `glue:TagResource` en sus políticas.

Cuando utilice etiquetas con AWS Glue, tenga en cuenta lo siguiente:

- Se permite un máximo de 50 etiquetas por entidad.
- En AWS Glue, debe especificar etiquetas como una lista de pares clave-valor con el formato `{"string": "string" ...}`
- Al crear una etiqueta en un objeto, la clave de etiqueta es obligatoria y el valor de la etiqueta es opcional.
- La clave y el valor de la etiqueta distinguen entre mayúsculas y minúsculas.
- La clave de etiqueta y el valor de etiqueta no deben contener el prefijo aws. No se permiten operaciones en este tipo de etiquetas.
- La longitud máxima de la clave de etiqueta es de 128 caracteres Unicode en UTF-8. La clave de etiqueta no debe estar vacía ni ser nula.
- La longitud máxima del valor de etiqueta es de 256 caracteres Unicode en UTF-8. El valor de etiqueta puede estar vacío o ser nulo.

Compatibilidad con el etiquetado en conexiones de AWS Glue

Puede restringir los permisos de la acción `CreateConnection`, `UpdateConnection`, `GetConnection` y `DeleteConnection` en función de la etiqueta de recurso. Esto permite implementar el control de acceso de privilegio mínimo en trabajos de AWS Glue con orígenes de datos JDBC que necesitan obtener información de conexión JDBC de Data Catalog.

Ejemplo de uso

Creación de una conexión AWS Glue con la etiqueta ["connection-category", "dev-test"].

Especifique la condición de etiqueta para la acción `GetConnection` en la política de IAM.

```
{
   "Effect": "Allow",
   "Action": [
     "glue:GetConnection"
```

```
    ],
   "Resource": "*",
   "Condition": {
     "ForAnyValue:StringEquals": {
       "aws:ResourceTag/tagKey": "dev-test"
     }
   }
 }
```

Ejemplos

Los siguientes ejemplos crean un trabajo con etiquetas asignadas.

AWS CLI

```
aws glue create-job --name job-test-tags --role MyJobRole --command
 Name=glueetl,ScriptLocation=S3://aws-glue-scripts//prod-job1
--tags key1=value1,key2=value2
```

AWS CloudFormation JSON

```
{
  "Description": "AWS Glue Job Test Tags",
  "Resources": {
    "MyJobRole": {
      "Type": "AWS::IAM::Role",
      "Properties": {
        "AssumeRolePolicyDocument": {
          "Version": "2012-10-17",
          "Statement": [
            {
              "Effect": "Allow",
              "Principal": {
                "Service": [
                  "glue.amazonaws.com"
                ]
              },
              "Action": [
                "sts:AssumeRole"
              ]
            }
          ]
        },
```

```
        "Path": "/",
        "Policies": [
          {
            "PolicyName": "root",
            "PolicyDocument": {
              "Version": "2012-10-17",
              "Statement": [
                {
                  "Effect": "Allow",
                  "Action": "*",
                  "Resource": "*"
                }
              ]
            }
          }
        ]
      }
    },
    "MyJob": {
      "Type": "AWS::Glue::Job",
      "Properties": {
        "Command": {
          "Name": "glueetl",
          "ScriptLocation": "s3://aws-glue-scripts//prod-job1"
        },
        "DefaultArguments": {
          "--job-bookmark-option": "job-bookmark-enable"
        },
        "ExecutionProperty": {
          "MaxConcurrentRuns": 2
        },
        "MaxRetries": 0,
        "Name": "cf-job1",
        "Role": {
           "Ref": "MyJobRole",
           "Tags": {
                "key1": "value1",
                "key2": "value2"
           }
        }
      }
    }
  }
```

```
}
```

AWS CloudFormation YAML

```
Description: AWS Glue Job Test Tags
Resources:
  MyJobRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - glue.amazonaws.com
            Action:
              - sts:AssumeRole
      Path: "/"
      Policies:
        - PolicyName: root
          PolicyDocument:
            Version: '2012-10-17'
            Statement:
              - Effect: Allow
                Action: "*"
                Resource: "*"
  MyJob:
    Type: AWS::Glue::Job
    Properties:
      Command:
        Name: glueetl
        ScriptLocation: s3://aws-glue-scripts//prod-job1
      DefaultArguments:
        "--job-bookmark-option": job-bookmark-enable
      ExecutionProperty:
        MaxConcurrentRuns: 2
      MaxRetries: 0
      Name: cf-job1
      Role:
        Ref: MyJobRole
        Tags:
           key1: value1
```

```
            key2: value2
```

Para obtener más información, consulte Estrategias de etiquetado de AWS.

Para obtener más información acerca de cómo controlar el acceso con etiquetas, consulte ABAC con Glue AWS.

Automatización de AWS Glue con CloudWatch Events

Puede utilizar Eventos de Amazon CloudWatch para automatizar los servicios de AWS y responder de forma automática a eventos del sistema, como problemas de disponibilidad de aplicaciones o cambios de recursos. Los eventos de los servicios de AWS se envían a CloudWatch Events casi en tiempo real. Puede crear reglas sencillas para indicar qué eventos le resultan de interés, así como qué acciones automatizadas se van a realizar cuando un evento cumple una de las reglas. Entre las acciones que se pueden activar automáticamente se incluyen las siguientes:

- Invocar una función de AWS Lambda
- Invocar Ejecutar comando de Amazon EC2
- Desviar el evento a Amazon Kinesis Data Streams
- Activar una máquina de estado de AWS Step Functions
- Notificar un tema de Amazon SNS o una cola de Amazon SQS

Algunos ejemplos del uso de CloudWatch Events con AWS Glue incluyen los siguientes:

- Activar una función de Lambda cuando se realiza correctamente un trabajo de ETL
- Notificar un tema de Amazon SNS cuando un trabajo de ETL produce error

Los siguientes eventos de CloudWatch Events son generados por AWS Glue.

- Se generan eventos de `"detail-type":"Glue Job State Change"` para `SUCCEEDED`, `FAILED`, `TIMEOUT` y `STOPPED`.
- Se generan eventos de `"detail-type":"Glue Job Run Status"` para ejecuciones de trabajos `RUNNING`, `STARTING` o `STOPPING` cuando superan el umbral de notificación de retraso de flujo de trabajo. Debe configurar la propiedad de umbral de notificación de retraso en el trabajo para recibir estos eventos.

Solo se genera un evento por estado de ejecución de un trabajo cuando se supera el umbral de notificación de retraso en el trabajo.

- Se generan eventos de `"detail-type":"Glue Crawler State Change"` para `Started`, `Succeeded` y `Failed`.
- Se generan eventos de `"detail-type":"Glue Data Catalog Database State Change"` para `CreateDatabase`, `DeleteDatabase`, `CreateTable`, `DeleteTable` y `BatchDeleteTable`. Por ejemplo, si se crea o se elimina una tabla, se envía una notificación a CloudWatch Events. Tenga en cuenta que no puede escribir un programa que dependa del orden o de la existencia de eventos de notificación, ya que pueden no estar ordenados o faltar. Los eventos se emiten en la medida de lo posible. En los detalles de la notificación:
  - `typeOfChange` contiene el nombre de una operación API.
  - `databaseName` contiene el nombre de la base de datos afectada.
  - `changedTables` contiene hasta 100 nombres de tablas afectadas por notificación. Cuando los nombres de las tablas son largos, podrían crearse varias notificaciones.
- Se generan eventos de `"detail-type":"Glue Data Catalog Table State Change"` para `UpdateTable`, `CreatePartition`, `BatchCreatePartition`, `UpdatePartition`, `DeletePartition`, `BatchUpdatePartition` y `BatchDeletePartition`. Por ejemplo, si se actualiza una tabla o una partición, se envía una notificación a CloudWatch Events. Tenga en cuenta que no puede escribir un programa que dependa del orden o de la existencia de eventos de notificación, ya que pueden no estar ordenados o faltar. Los eventos se emiten en la medida de lo posible. En los detalles de la notificación:
  - `typeOfChange` contiene el nombre de una operación API.
  - `databaseName` contiene el nombre de la base de datos que contiene los recursos afectados.
  - `tableName` contiene el nombre de la tabla afectada.
  - `changedPartitions` especifica hasta 100 particiones afectadas en una notificación. Cuando los nombres de las particiones son largos, podrían crearse varias notificaciones.

    Por ejemplo, si hay dos claves de partición, `Year` y `Month`, `"2018,01", "2018,02"` modifica la partición donde `"Year=2018" and "Month=01"` y la partición donde `"Year=2018" and "Month=02"`.

```
{
    "version":"0",
    "id":"abcdef00-1234-5678-9abc-def012345678",
    "detail-type":"Glue Data Catalog Table State Change",
```

```
    "source":"aws.glue",
    "account":"123456789012",
    "time":"2017-09-07T18:57:21Z",
    "region":"us-west-2",
    "resources":["arn:aws:glue:us-west-2:123456789012:database/default/foo"],
    "detail":{
        "changedPartitions": [
            "2018,01",
            "2018,02"
        ],
        "databaseName": "default",
        "tableName": "foo",
        "typeOfChange": "BatchCreatePartition"
        }
}
```

Para obtener más información, consulte la Guía del usuario de los Eventos de Amazon CloudWatch. Para eventos específicos de AWS Glue consulte Eventos de AWS Glue.

Monitoreo de recursos AWS Glue

AWS Glue tiene límites de servicio para proteger a los clientes de un aprovisionamiento excesivo inesperado y de acciones malintencionadas destinadas a aumentar la factura. Los límites también protegen el servicio. Al iniciar sesión en la consola de Service Quotas de AWS, los clientes pueden ver sus límites de recursos actuales y solicitar un aumento (si procede).

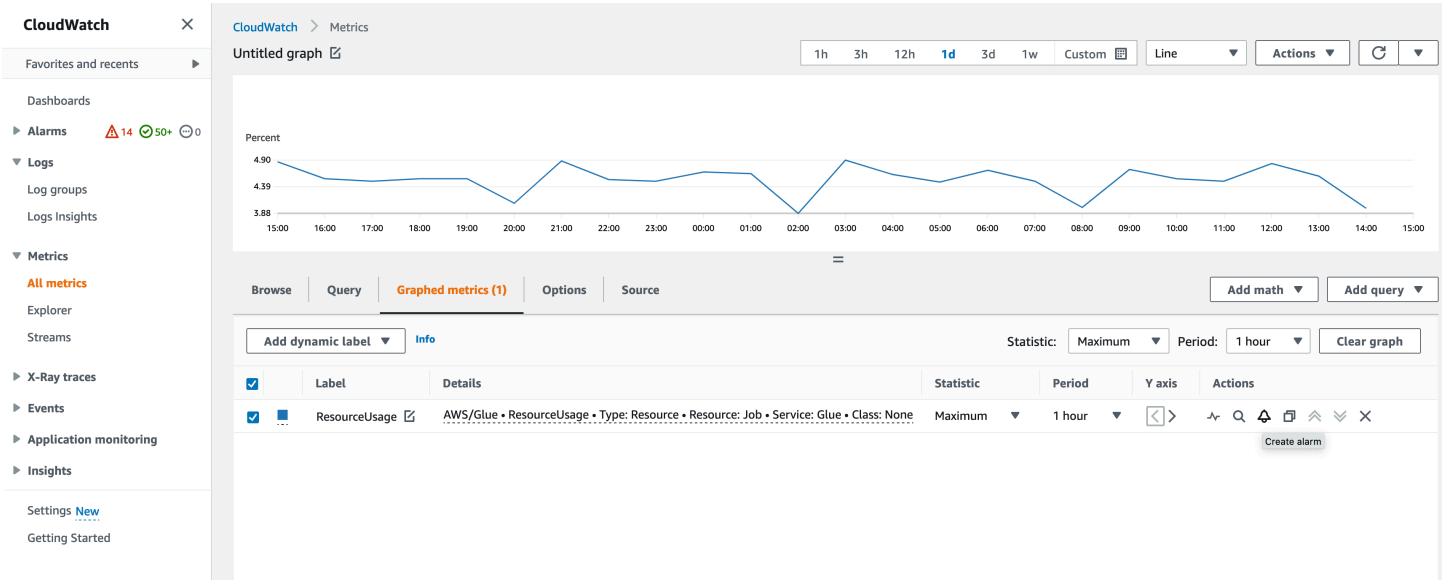
AWS Glue permite ver el uso de los recursos del servicio como un porcentaje en Amazon CloudWatch y configurar las alarmas de CloudWatch para supervisar el uso. Amazon CloudWatch supervisa los recursos de AWS y las aplicaciones de los clientes que se ejecutan en la infraestructura de Amazon. Las métricas son gratuitas para usted. Las siguientes métricas son compatibles:

- Número de flujos de trabajo por cuenta
- Número de disparadores por cuenta
- Número de trabajos por cuenta
- Número de ejecuciones de flujo de trabajo simultáneas por cuenta
- Número de esquemas por cuenta
- Número de sesiones interactivas por cuenta

Configuración y uso de métricas de recursos

Para utilizar esta característica, puede ir a la consola de Amazon CloudWatch para ver las métricas y configurar las alarmas. Las métricas se encuentran en el espacio de nombres AWS/Glue y son un porcentaje del recuento real de uso de recursos dividido por la cuota de recursos. Las métricas de CloudWatch se envían a sus cuentas, lo que no supondrá ningún costo. Por ejemplo, si ha creado 10 flujos de trabajo y la cuota de servicio permite tener 200 flujos de trabajo como máximo, el uso es de 10/200 = 5 % y, en el gráfico, verá un punto de datos de 5 como porcentaje. Para ser más específicos:

```
Namespace: AWS/Glue
Metric name: ResourceUsage
Type: Resource
Resource: Workflow (or Trigger, Job, JobRun, Blueprint, InteractiveSession)
Service: Glue
Class: None
```



Creación de una alarma en una métrica en la consola de CloudWatch:

1. Una vez que haya localizado la métrica, vaya a Métricas gráficas.
2. Haga clic en Crear alarma en Acciones.
3. Configure la alarma según sea necesario.

Emitimos métricas cada vez que se produce un cambio en el uso de los recursos (por ejemplo, un aumento o una disminución). Sin embargo, si el uso de sus recursos no cambia, emitimos métricas

cada hora para que tenga un gráfico continuo de CloudWatch. Para evitar que falten puntos de datos, no recomendamos que configure un periodo inferior a 1 hora.

También puede configurar las alarmas con AWS CloudFormation, tal como en el siguiente ejemplo. En este ejemplo, cuando el uso de los recursos del flujo de trabajo alcanza el 80 %, activa una alarma para enviar un mensaje al tema de SNS existente, donde puede suscribirse para recibir notificaciones.

```
{
 "Type": "AWS::CloudWatch::Alarm",
 "Properties": {
  "AlarmName": "WorkflowUsageAlarm",
  "ActionsEnabled": true,
  "OKActions": [],
  "AlarmActions": [
   "arn:aws:sns:af-south-1:085425700061:Default_CloudWatch_Alarms_Topic"
  ],
  "InsufficientDataActions": [],
  "MetricName": "ResourceUsage",
  "Namespace": "AWS/Glue",
  "Statistic": "Maximum",
  "Dimensions": [{
    "Name": "Type",
    "Value": "Resource"
   },
   {
    "Name": "Resource",
    "Value": "Workflow"
   },
   {
    "Name": "Service",
    "Value": "Glue"
   },
   {
    "Name": "Class",
    "Value": "None"
   }
  ],
  "Period": 3600,
  "EvaluationPeriods": 1,
  "DatapointsToAlarm": 1,
  "Threshold": 80,
  "ComparisonOperator": "GreaterThanThreshold",
```

```
  "TreatMissingData": "notBreaching"
 }
}
```

Registro de llamadas a la API de AWS Glue con AWS CloudTrail

AWS Glue se integra con AWS CloudTrail, un servicio que proporciona un registro de las acciones hechas por un usuario, un rol o un servicio de AWS en AWS Glue. CloudTrail captura todas las llamadas a la API de AWS Glue como eventos. Las llamadas capturadas incluyen las llamadas desde la consola de AWS Glue y las llamadas desde el código a las operaciones de la API de AWS Glue. Si crea un registro de seguimiento, puede habilitar la entrega continua de eventos de CloudTrail a un bucket de Amazon S3, incluidos los eventos para AWS Glue. Si no configura un registro de seguimiento, puede ver los eventos más recientes en la consola de CloudTrail en el Historial de eventos. Mediante la información recopilada por CloudTrail, puede determinar la solicitud que se realizó a AWS Glue, la dirección IP desde la que se realizó, quién la realizó y cuándo, etc.

Para más información sobre CloudTrail, consulte la Guía del usuario de AWS CloudTrail.

Información de AWS Glue en CloudTrail

CloudTrail se habilita en su cuenta de AWS cuando la crea. Cuando se produce una actividad en AWS Glue, esa actividad se registra en un evento de CloudTrail junto con otros eventos de servicio de AWS en Historial de eventos. Puede ver, buscar y descargar los últimos eventos de la cuenta de AWS. Para obtener más información, consulte Visualización de eventos con el historial de eventos de CloudTrail.

Para mantener un registro continuo de eventos en la cuenta de AWS, incluidos los eventos de AWS Glue, cree un registro de seguimiento. Un registro de seguimiento permite a CloudTrail enviar archivos de registro a un bucket de Amazon S3. De manera predeterminada, cuando se crea un registro de seguimiento en la consola, el registro de seguimiento se aplica a todas las regiones de AWS. El registro de seguimiento registra los eventos de todas las regiones de la partición de AWS y envía los archivos de registro al bucket de Amazon S3 especificado. También es posible configurar otros servicios de AWS para analizar en profundidad y actuar en función de los datos de eventos recopilados en los registros de CloudTrail. Para más información, consulte los siguientes temas:

- Creación de un seguimiento para su cuenta de AWS
- Servicios e integraciones compatibles con CloudTrail
- Configurar notificaciones de Amazon SNS para CloudTrail

- Recepción de archivos de registro de CloudTrail de varias regiones y Recepción de archivos de registro de CloudTrail de varias cuentas

CloudTrail registra todas las acciones de AWS Glue y se documentan en AWS Glue API. Por ejemplo, las llamadas a las acciones `CreateDatabase`, `CreateTable` y `CreateScript` generan entradas en los archivos de registros de CloudTrail.

Cada entrada de registro o evento contiene información sobre quién generó la solicitud. La información de identidad del usuario le ayuda a determinar lo siguiente:

- Si la solicitud se realizó con las credenciales raíz o del usuario de IAM.
- Si la solicitud se realizó con credenciales de seguridad temporales de un rol o fue un usuario federado.
- Si la solicitud la realizó otro servicio de AWS.

Para obtener más información, consulte el Elemento userIdentity de CloudTrail.

Sin embargo, CloudTrail no registra toda la información referente a las llamadas. Por ejemplo, no registra determinada información confidencial como `ConnectionProperties`, que se utiliza en las solicitudes de conexión, y registra `null` en lugar de las respuestas que envían las API siguientes:

```
BatchGetPartition       GetCrawlers          GetJobs          GetTable
CreateScript            GetCrawlerMetrics    GetJobRun        GetTables
GetCatalogImportStatus  GetDatabase          GetJobRuns       GetTableVersions
GetClassifier           GetDatabases         GetMapping       GetTrigger
GetClassifiers          GetDataflowGraph     GetObjects       GetTriggers
GetConnection           GetDevEndpoint       GetPartition     GetUserDefinedFunction
GetConnections          GetDevEndpoints      GetPartitions    GetUserDefinedFunctions
GetCrawler              GetJob               GetPlan
```

Descripción de las entradas de los archivos de registro de AWS Glue

Un registro de seguimiento es una configuración que permite la entrega de eventos como archivos de registro a un bucket de Amazon S3 que especifique. Los archivos de registro de CloudTrail pueden contener una o varias entradas de registro. Un evento representa una solicitud específica realizada desde un origen cualquiera, y contiene información sobre la acción solicitada, la fecha y la hora de la acción, los parámetros de la solicitud, etc. Los archivos de registro de CloudTrail no rastrean el orden en la pila de las llamadas públicas a la API, por lo que estas no aparecen en ningún orden específico.

En el siguiente ejemplo, se muestra una entrada de registro de CloudTrail que ilustra la acción `DeleteCrawler`.


```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AKIAIOSFODNN7EXAMPLE",
    "arn": "arn:aws:iam::123456789012:user/johndoe",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName": "johndoe"
  },
  "eventTime": "2017-10-11T22:29:49Z",
  "eventSource": "glue.amazonaws.com",
  "eventName": "DeleteCrawler",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "72.21.198.64",
  "userAgent": "aws-cli/1.11.148 Python/3.6.1 Darwin/16.7.0 botocore/1.7.6",
  "requestParameters": {
    "name": "tes-alpha"
  },
  "responseElements": null,
  "requestID": "b16f4050-aed3-11e7-b0b3-75564a46954f",
  "eventID": "e73dd117-cfd1-47d1-9e2f-d1271cad838c",
  "eventType": "AwsApiCall",
  "recipientAccountId": "123456789012"
}
```


Este ejemplo muestra una entrada de registro de CloudTrail que ilustra la acción `CreateConnection`.


```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AKIAIOSFODNN7EXAMPLE",
    "arn": "arn:aws:iam::123456789012:user/johndoe",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName": "johndoe"
  },
  "eventTime": "2017-10-13T00:19:19Z",
```

```
  "eventSource": "glue.amazonaws.com",
  "eventName": "CreateConnection",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "72.21.198.66",
  "userAgent": "aws-cli/1.11.148 Python/3.6.1 Darwin/16.7.0 botocore/1.7.6",
  "requestParameters": {
    "connectionInput": {
      "name": "test-connection-alpha",
      "connectionType": "JDBC",
      "physicalConnectionRequirements": {
        "subnetId": "subnet-323232",
        "availabilityZone": "us-east-1a",
        "securityGroupIdList": [
          "sg-12121212"
        ]
      }
    }
  },
  "responseElements": null,
  "requestID": "27136ebc-afac-11e7-a7d6-ab217e5c3f19",
  "eventID": "e8b3baeb-c511-4597-880f-c16210c60a4a",
  "eventType": "AwsApiCall",
  "recipientAccountId": "123456789012"
}
```


Estados de ejecución de trabajos de AWS Glue

Puede ver el estado de un trabajo del servicio ETL (extracción, transformación y carga) de AWS Glue mientras se está ejecutando o después de su detención. Puede ver el estado mediante la consola de AWS Glue, la AWS Command Line Interface (AWS CLI), o la acción GetJobRun en la API de AWS Glue.

Los estados de ejecuciones de trabajo posibles son `STARTING`, `RUNNING`, `STOPPING`, `STOPPED`, `SUCCEEDED`, `FAILED`, `ERROR`, `WAITING` y `TIMEOUT`.

En la tabla siguiente se muestran los estados que indican la terminación anormal de la tarea.

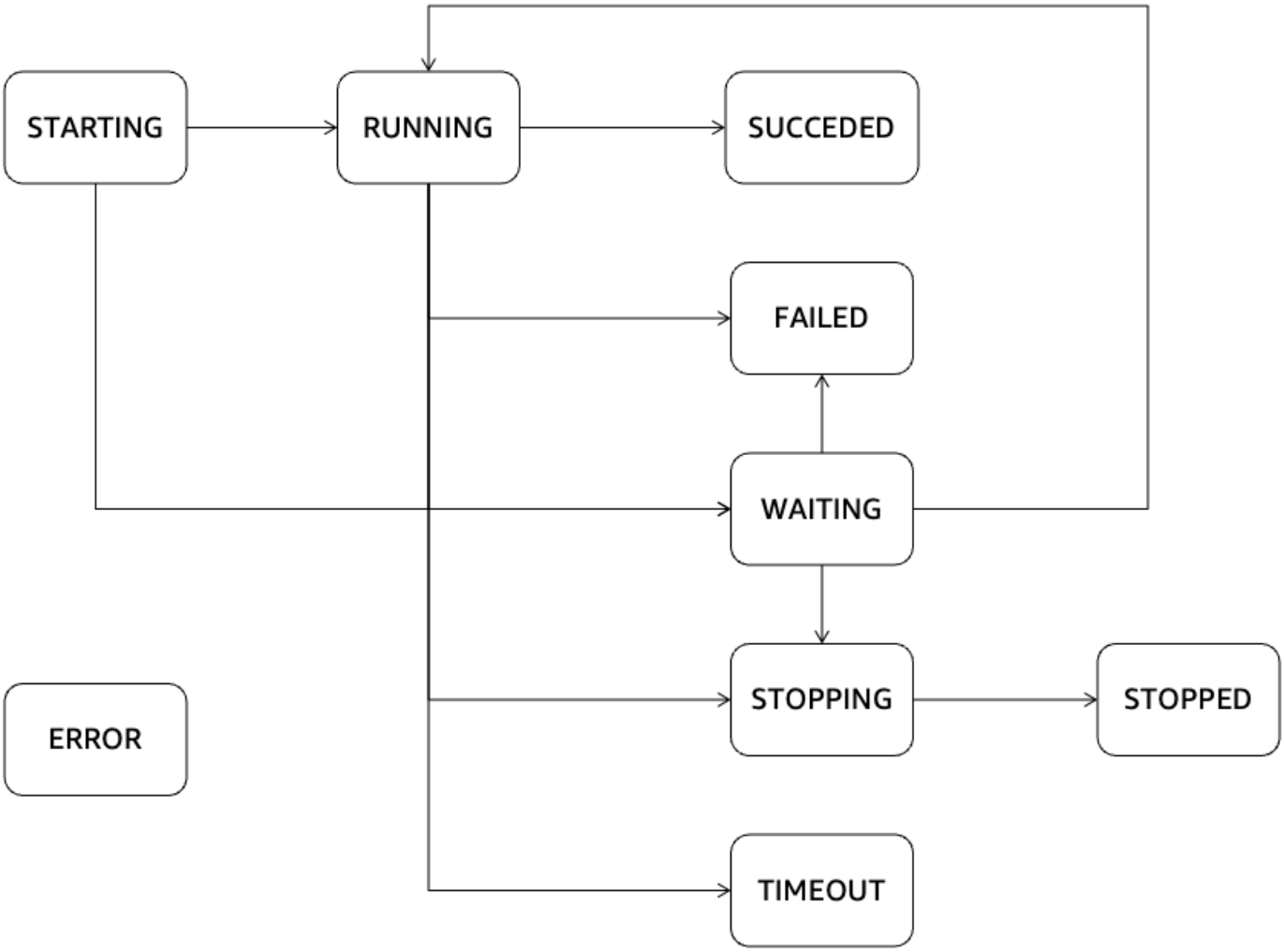
Estados de ejecuciones de trabajos	Descripción
`FAILED`	El trabajo superó el máximo de ejecuciones simultáneas permitidas o finalizó con un código de salida desconocido.
`ERROR`	Un flujo de trabajo, un desencadenador de programación o un desencadenador de eventos intentó ejecutar un trabajo eliminado.
`TIMEOUT`	El tiempo de ejecución del trabajo superó el valor de tiempo de espera especificado.

El estado `WAITING` indica que una ejecución de trabajo está esperando recursos. En la siguiente tabla se describe el comportamiento de espera para diferentes clases de trabajos.

Tipo de trabajo	Comportamiento
Trabajos de Spark (estándar)	Los trabajos que no se hayan configurado para volver a intentarlo en función de su configura ción de `maxRetries` pueden entrar en el estado DE ESPERA. Una nueva ejecución de trabajo estará en estado EN ESPERA si el servicio no puede adquirir recursos suficientes para iniciar la ejecución. Esto puede deberse a que las cuotas de servicio de su cuenta o a los límites de capacidad de su región se produzcan en alguno de los siguientes casos de error: • Se ha superado el número máximo de ejecuciones simultáneas por cuenta • Se ha superado el número máximo de ejecuciones simultáneas por trabajo (incluye la cuota de servicio a nivel de cuenta, así como el límite que especifique en el trabajo con `MaxConcurrentRuns`)

Tipo de trabajo	Comportamiento
	• Se ha superado el número máximo de cómputo simultáneo (uso de DPU) • Recurso no disponible Para obtener más información sobre las cuotas de servicio de AWS Glue, consulte puntos de conexión y cuotas de AWS Glue. El tiempo que AWS Glue esperará para recibir los recursos puede variar según las circunstancias. Un trabajo puede pasar de un estado no terminal a otro al intentar adquirir recursos. Eventualm ente, el trabajo pasará a ser `FAILED` si no puede adquirir recursos. AWS Glue volverá a intentarlo durante un máximo de 15 minutos o 10 intentos, lo que ocurra primero.
Trabajos de Spark (Flex)	Una nueva ejecución de trabajo estará en estado EN ESPERA si el servicio no puede adquirir recursos suficientes para iniciar la ejecución, lo que retrasa el inicio de la ejecución. La ejecución estará en estado EN ESPERA durante un máximo de 20 minutos (tiempo de espera controlado por el servicio) . Después de 15 minutos, el servicio intentará forzar el inicio y, según la capacidad disponibl e, la ejecución puede comenzar o fallar con el mensaje de error correspondiente.
Trabajos de shell de Python	El mismo comportamiento que los trabajos estándar con Spark.

El diagrama que se muestra a continuación describe el estado esperado de las transiciones a través del ciclo de vida de un trabajo de AWS Glue. Esta información se aplica a todos los tipos de trabajo.


STARTING
RUNNING
SUCCEDED
FAILED
WAITING
STOPPING
STOPPED
ERROR
TIMEOUT

AWS Glue Streaming

AWS Glue La transmisión, un componente de AWS Glue, le permite gestionar de forma eficiente los datos de streaming prácticamente en tiempo real, lo que le permite llevar a cabo tareas cruciales como la ingesta de datos, el procesamiento y el aprendizaje automático. Al utilizar el marco de transmisión Apache Spark, AWS Glue Streaming proporciona un servicio sin servidor que puede gestionar datos de transmisión a gran escala. AWS Glue proporciona varias optimizaciones además de Apache Spark, como la infraestructura sin servidor, el autoscalamiento, el desarrollo visual de tareas, libretas instantáneas para tareas de streaming y otras mejoras de rendimiento.

Casos de uso de la transmisión

Algunos casos de uso comunes del streaming incluyen: AWS Glue

Sin procesamiento de ear-real-time datos: la AWS Glue transmisión permite a las organizaciones procesar los datos de transmisión casi en tiempo real, lo que les permite obtener información y tomar decisiones oportunas en función de la información más reciente.

Detección de fraudes: puede utilizar AWS Glue Streaming para analizar en tiempo real los datos de streaming, lo que lo convierte en una herramienta valiosa para detectar actividades fraudulentas, como el fraude con tarjetas de crédito, la intrusión en la red o las estafas en línea. Al procesar y analizar continuamente los datos entrantes, puede identificar rápidamente patrones o anomalías sospechosos.

Análisis de redes sociales: el AWS Glue streaming puede procesar datos de redes sociales en tiempo real, como tuits, publicaciones o comentarios, lo que permite a las organizaciones monitorear las tendencias, analizar las opiniones y gestionar la reputación de la marca en tiempo real.

Análisis de Internet de las cosas (IoT): la AWS Glue transmisión es adecuada para gestionar y analizar flujos de datos a alta velocidad generados por dispositivos, sensores y maquinaria conectada de IoT. Permite la supervisión en tiempo real, la detección de anomalías, el mantenimiento predictivo y otros casos de uso de análisis de IoT.

Análisis del flujo de clics: la AWS Glue transmisión puede procesar y analizar datos del flujo de clics en tiempo real procedentes de sitios web o aplicaciones móviles. Esto permite a las empresas obtener información acerca del comportamiento de los usuarios, personalizar las experiencias de los usuarios y optimizar las campañas de marketing según los datos del flujo de clics en tiempo real.

Supervisión y análisis de registros: AWS Glue Streaming puede procesar y analizar continuamente los datos de registro de servidores, aplicaciones o dispositivos de red en tiempo real. Esto ayuda a detectar anomalías, solucionar problemas y supervisar el estado y el rendimiento del sistema.

Sistemas de recomendaciones: La AWS Glue transmisión puede procesar los datos de actividad de los usuarios en tiempo real y actualizar los modelos de recomendación de forma dinámica. Esto permite realizar recomendaciones personalizadas y en tiempo real en función del comportamiento y las preferencias de los usuarios.

Estos son algunos ejemplos de la amplia gama de casos de uso en los que se puede aplicar el AWS Glue streaming. Su integración con el AWS ecosistema y los servicios gestionados lo convierten en una opción práctica para el procesamiento y el análisis de transmisiones en tiempo real en la nube.

¿Cuáles son las ventajas de usar AWS Glue Streaming?

Los beneficios de usar el AWS Glue streaming son los siguientes:

- Sin servidor: la AWS Glue transmisión no tiene servidores, lo que elimina la necesidad de administrar la infraestructura. Esto reduce la sobrecarga operativa y permite a los usuarios centrarse en el procesamiento de datos y las tareas de análisis en lugar de la administración de la infraestructura.
- Escalado automático: la AWS Glue transmisión proporciona capacidades de escalado automático, ajustando dinámicamente la capacidad de procesamiento en función de la carga de trabajo. Se escala o reduce horizontalmente de manera automática para gestionar las fluctuaciones en el volumen de datos, lo que garantiza un rendimiento y un uso de los recursos óptimos.
- Desarrollo visual: el desarrollo de puestos de trabajo en streaming puede resultar complejo. AWS Glue Streaming aborda este desafío al ofrecer AWS Glue Studio, una herramienta de creación visual. AWS Glue Studio simplifica el proceso de creación de flujos de trabajo de streaming y permite a los desarrolladores diseñar y gestionar aplicaciones de streaming de forma visual, lo que reduce la curva de aprendizaje y aumenta la productividad.
- Rentable: al ser un servicio sin servidor, AWS Glue Streaming ofrece rentabilidad al eliminar la necesidad de aprovisionar y mantener la infraestructura. A los usuarios se les factura en función de los recursos consumidos durante la ejecución de los trabajos de transmisión, lo que permite optimizar los costos y escalarlos en función del uso real.
- Gestiona cargas de trabajo complejas: la AWS Glue transmisión está diseñada para gestionar cargas de trabajo de streaming complejas. Puede procesar y analizar grandes volúmenes de datos

en tiempo real, soportar transformaciones avanzadas e integrarse con otros AWS servicios, lo que permite flujos de datos de streaming y flujos de trabajo analíticos sofisticados.

- Sin dependencia: la AWS Glue transmisión proporciona flexibilidad y evita la dependencia de un proveedor. Los usuarios pueden aprovechar el AWS Glue streaming como parte de un AWS ecosistema más amplio, integrándolo con otros servicios sin problemas. AWS Esto permite una fácil integración con los orígenes de datos, las aplicaciones y los servicios existentes sin estar atados a una tecnología o plataforma específica.

¿Cuándo usar AWS Glue Streaming?

Hay muchas opciones en lo que respecta a los casos de uso de la transmisión. Recomendamos hacer AWS Glue streaming en los siguientes escenarios.

1. Si ya utilizas AWS Glue Spark para el procesamiento por lotes, AWS Glue Streaming es la opción ideal para ti. Proporciona una transición perfecta a la creación de trabajos de transmisión sin necesidad de aprender un nuevo lenguaje o marco. Al aprovechar los conocimientos y la infraestructura existentes, AWS Glue Streaming simplifica el proceso de desarrollo del trabajo y te permite ampliar fácilmente tus capacidades de procesamiento de datos a escenarios de streaming en tiempo real.
2. Si necesita un servicio o producto unificado para gestionar las cargas de trabajo por lotes, de streaming y basadas en eventos, AWS Glue Streaming es la solución para usted. Con AWS Glue Streaming, puede consolidar sus necesidades de procesamiento de datos en un único marco, lo que elimina la complejidad de administrar varios sistemas. Esto permite el desarrollo y el mantenimiento eficientes de diversos flujos de trabajo de datos y, al mismo tiempo, garantiza la coherencia y la compatibilidad entre los diferentes tipos de carga de trabajo.
3. AWS Glue La transmisión es ideal para escenarios que implican volúmenes de datos de transmisión extremadamente grandes y transformaciones complejas, como las uniones entre transmisiones o bases de datos relacionales. Puede procesar y analizar flujos de datos masivos de manera eficiente, lo que le permite abordar cargas de trabajo exigentes con facilidad. Tanto si se trata de una ingesta de datos a alta velocidad como de una compleja manipulación de datos, la escalabilidad y las capacidades de procesamiento avanzadas de AWS Glue Streaming garantizan un rendimiento óptimo y unos resultados precisos.
4. Si prefieres un enfoque visual para crear trabajos de streaming, te AWS Glue ofrece AWS Glue Studio, con el que podrás diseñar y gestionar visualmente tus aplicaciones de streaming, simplificando el proceso de desarrollo. Esta interfaz intuitiva permite a los desarrolladores crear,

configurar y supervisar los flujos de trabajo de transmisión mediante una interfaz visual, lo que reduce la curva de aprendizaje y aumenta la productividad.

5. AWS Glue El streaming es una opción excelente para los casos de near-real-time uso en los que existen acuerdos de nivel de servicio (SLA) estrictos que duran más de 10 segundos.
6. Si está creando un lago de datos transaccional con Apache Iceberg, Apache Hudi o Delta Lake, AWS Glue Streaming ofrece soporte nativo para estos formatos de tablas abiertas. Esta perfecta integración le permite procesar los datos de transmisión directamente desde estos lagos de datos transaccionales, lo que garantiza la coherencia, integridad y compatibilidad de los datos.
7. Cuando necesite ingerir datos de streaming para una variedad de destinos de datos: AWS Glue Streaming proporciona destinos nativos a una variedad de destinos de datos como Amazon Redshift, Amazon RDS, Amazon Aurora, Oracle, SQL Server y otros destinos.

Orígenes de datos admitidos

AWS Glue La transmisión admite las siguientes fuentes de datos:

- Amazon Kinesis
- Amazon MSK (Managed Streaming para Apache Kafka)
- Apache Kafka autoadministrado

Destinos de datos admitidos

AWS Glue La transmisión admite una variedad de destinos de datos, tales como:

- Destinos de datos compatibles con AWS Glue Data Catalog
- Amazon S3
- Amazon Redshift
- MySQL
- PostgreSQL
- Oracle
- Microsoft SQL Server
- Snowflake
- Cualquier base de datos a la que se pueda conectar mediante JDBC

- Apache Iceberg, Delta y Apache Hudi
- AWS Glue Conectores Marketplace

Tutorial: Cree su primera carga de trabajo de transmisión con AWS Glue Studio

En este tutorial, aprenderá a crear un trabajo de transmisión con AWS Glue Studio. AWS Glue Studio es una interfaz visual para crear trabajos de AWS Glue.

Puede crear trabajos de extracción, transformación y carga (ETL) de transmisión que se ejecuten de forma continua y consuman datos de los orígenes de transmisión como Amazon Kinesis Data Streams, Apache Kafka y Amazon Managed Streaming para Apache Kafka (Amazon MSK).

Requisitos previos

Para seguir este tutorial, necesitará un usuario con permisos de consola de AWS para usar AWS Glue, Amazon Kinesis, Amazon S3, Amazon Athena, AWS CloudFormation, AWS Lambda y Amazon Cognito.

Consumir datos de transmisión desde Amazon Kinesis

Temas

- Generación de datos simulados con Kinesis Data Generator
- Creación de un trabajo de transmisión de AWS Glue con AWS Glue Studio
- Transformación y almacenamiento del resultado transformado en Amazon S3

Generación de datos simulados con Kinesis Data Generator

Puede generar sintéticamente datos de muestra en formato JSON mediante Kinesis Data Generator (KDG). Encontrará instrucciones y detalles completos en la documentación de la herramienta.

1. Para empezar, haga clic en

   para ejecutar una plantilla de AWS CloudFormation en su entorno de AWS.

> **Note**
>
> Es posible que se produzca un error en la plantilla de CloudFormation porque algunos recursos, como el usuario de Amazon Cognito para Kinesis Data Generator, ya existen en su cuenta de AWS. Esto puede deberse a que ya la configuró en otro tutorial o blog. Para solucionar este problema, puede probar la plantilla en una cuenta de AWS nueva para empezar de cero o explorar una región de AWS diferente. Estas opciones le permiten ejecutar el tutorial sin que entre en conflicto con los recursos existentes.

La plantilla aprovisiona un flujo de datos de Kinesis y una cuenta de Kinesis Data Generator. También crea un bucket de Amazon S3 para almacenar los datos y un rol de servicio de Glue con los permisos necesarios para este tutorial.

2. Introduzca un nombre de usuario y una contraseña que KDG usará para autenticarse. Tome nota del nombre de usuario y la contraseña para su uso posterior.
3. Seleccione Siguiente hasta el último paso. Acepte la creación de recursos de IAM. Compruebe si hay algún error en la parte superior de la pantalla, como que la contraseña no cumple los requisitos mínimos, e implemente la plantilla.
4. Navegue hasta la pestaña Salidas de la pila. Una vez implementada la plantilla, mostrará la propiedad generada KinesisDataGeneratorUrl. Haga clic en esa URL.
5. Ingrese el nombre de usuario y la contraseña que anotó.
6. Seleccione la región que está usando y elija Kinesis Stream `GlueStreamTest-{AWS::AccountId}`.
7. Escriba la siguiente plantilla:

```
{
    "ventilatorid": {{random.number(100)}},
    "eventtime": "{{date.now("YYYY-MM-DD HH:mm:ss")}}",
    "serialnumber": "{{random.uuid}}",
    "pressurecontrol": {{random.number(
        {
            "min":5,
            "max":30
        }
    )}},
    "o2stats": {{random.number(
        {
```

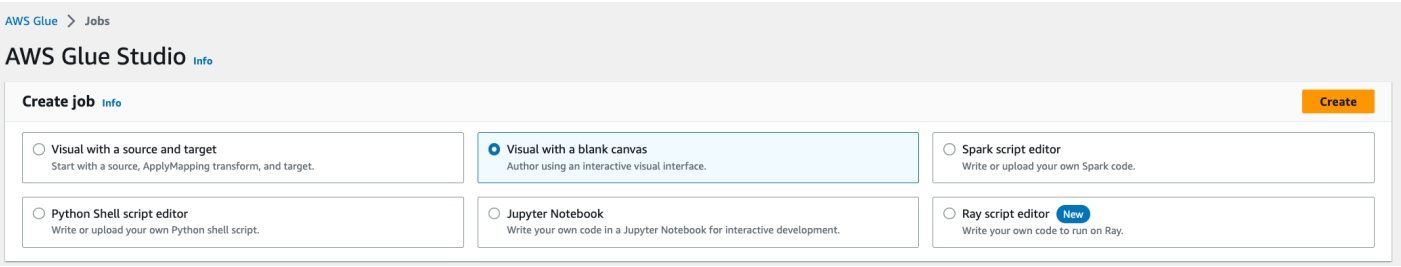
```
            "min":92,
            "max":98
        }
    )}},
    "minutevolume": {{random.number(
        {
            "min":5,
            "max":8
        }
    )}},
    "manufacturer": "{{random.arrayElement(
        ["3M", "GE","Vyaire", "Getinge"]
    )}}"
}
```

Ahora puede ver los datos simulados con la plantilla de prueba e incorporarlos a Kinesis con Enviar datos.

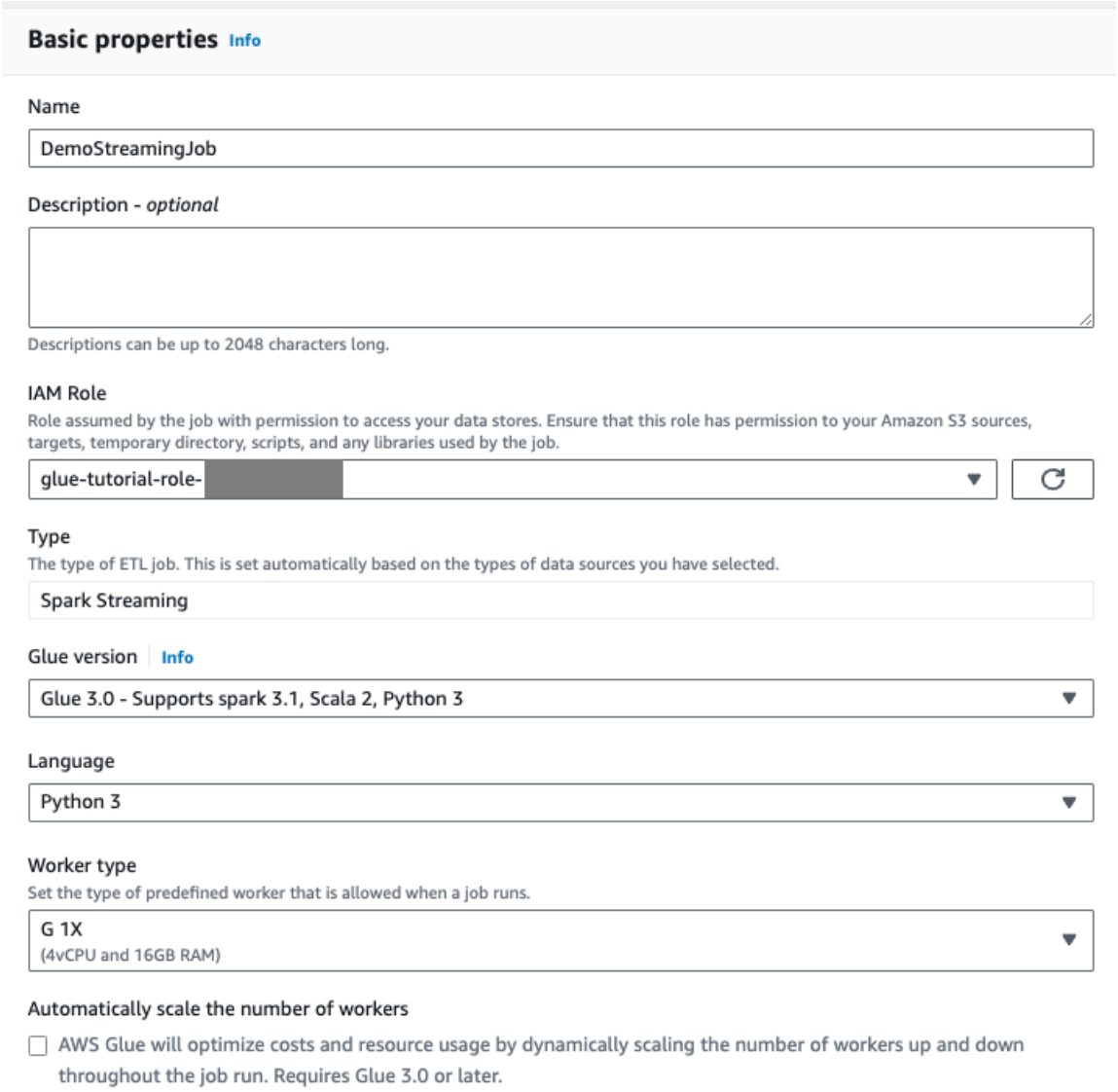
8. Haga clic en Enviar datos y genere entre 5 y 10 000 registros en Kinesis.

Creación de un trabajo de transmisión de AWS Glue con AWS Glue Studio

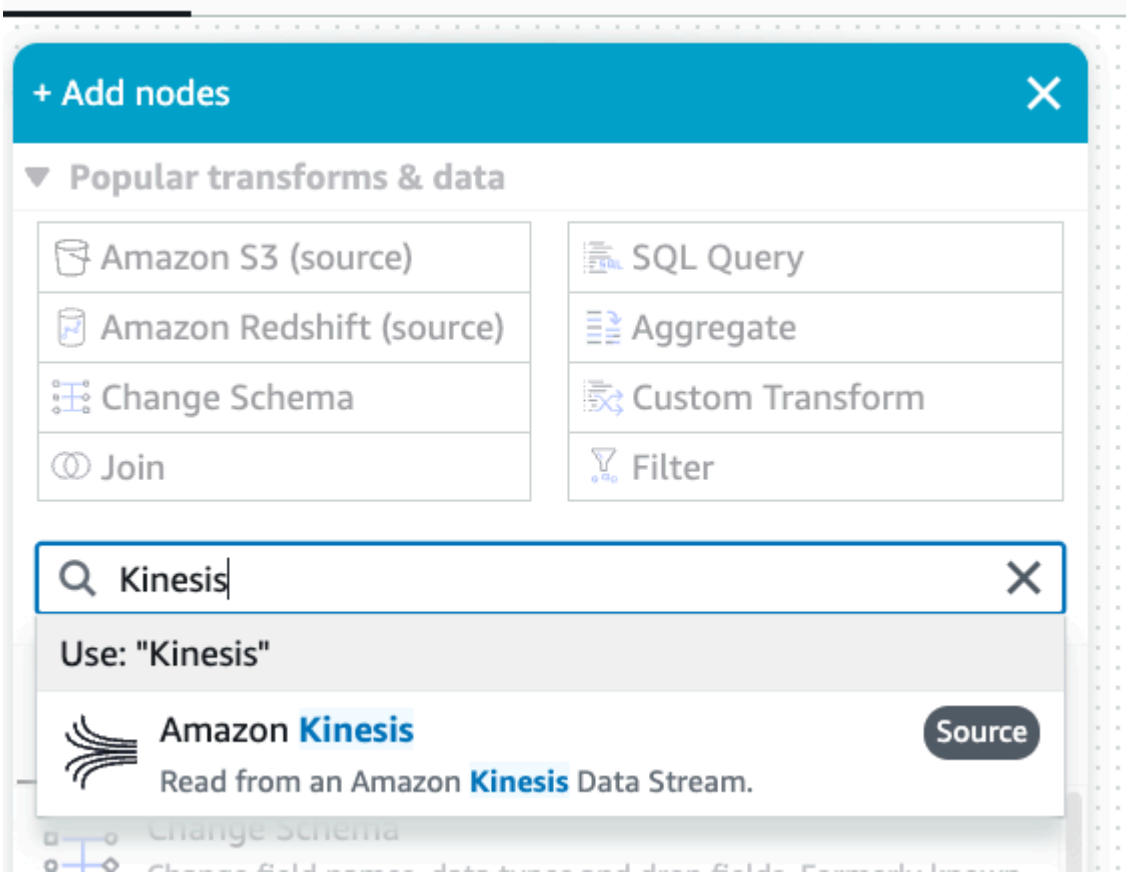
1. Navegue hasta AWS Glue en la consola en la misma región.
2. Seleccione Trabajos de ETL en la barra de navegación del lado izquierdo, en Integración de datos y ETL.
3. Cree un trabajo de AWS Glue a través de Visual con un lienzo en blanco.



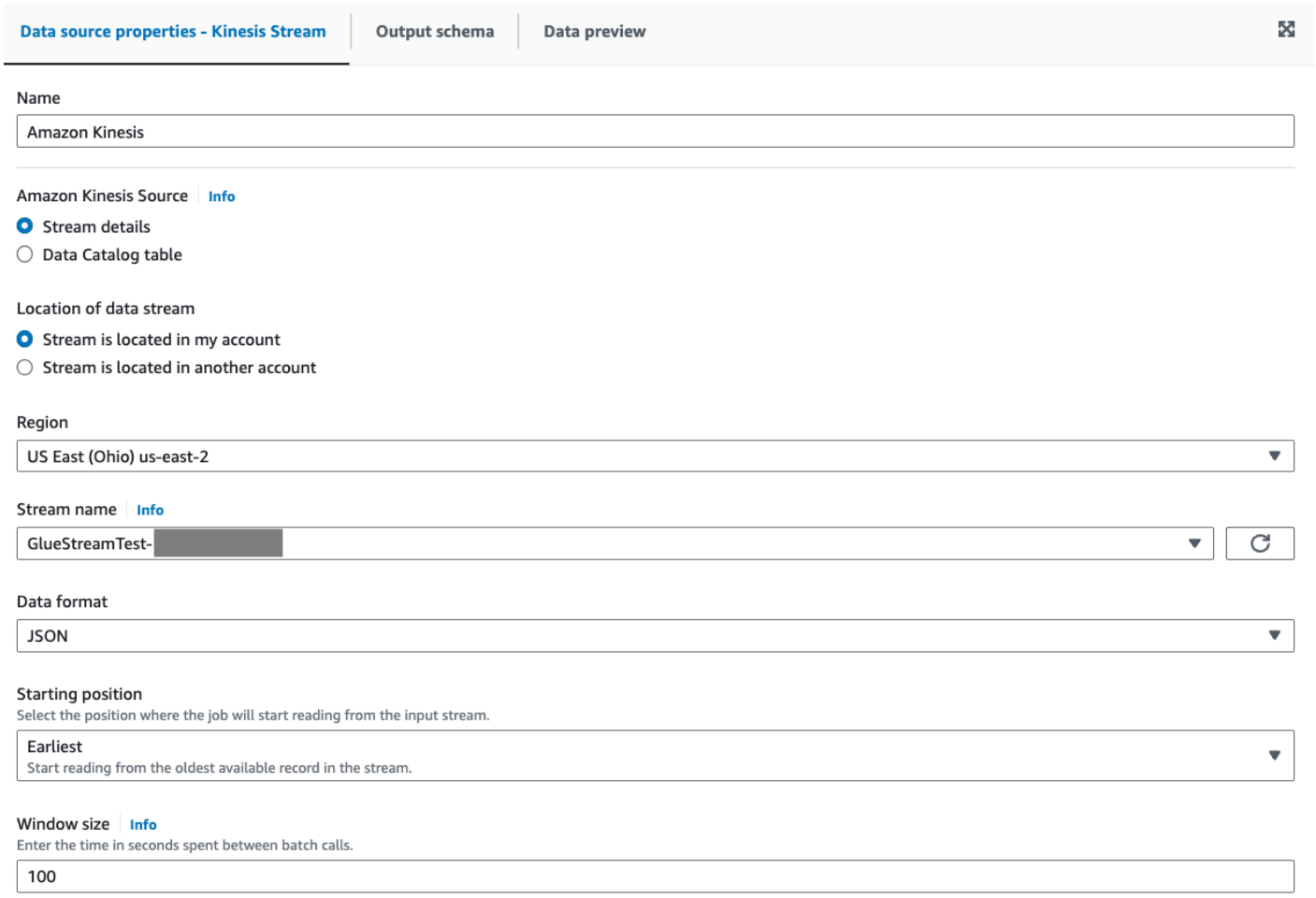
4. Vaya a la pestaña Detalles del trabajo.
5. Para el nombre del trabajo de AWS Glue, escriba `DemoStreamingJob`.
6. Para el rol de IAM, seleccione el rol aprovisionado en la plantilla de CloudFormation, `glue-tutorial-role-${AWS::AccountId}`.
7. En Versión de Glue, seleccione Glue 3.0. Deje todas las demás opciones en sus valores predeterminados.



8. Navegue hasta la pestaña Visual.
9. Haga clic en el icono del signo más. Escriba Kinesis en la barra de búsqueda. Seleccione el origen de datos de Amazon Kinesis.



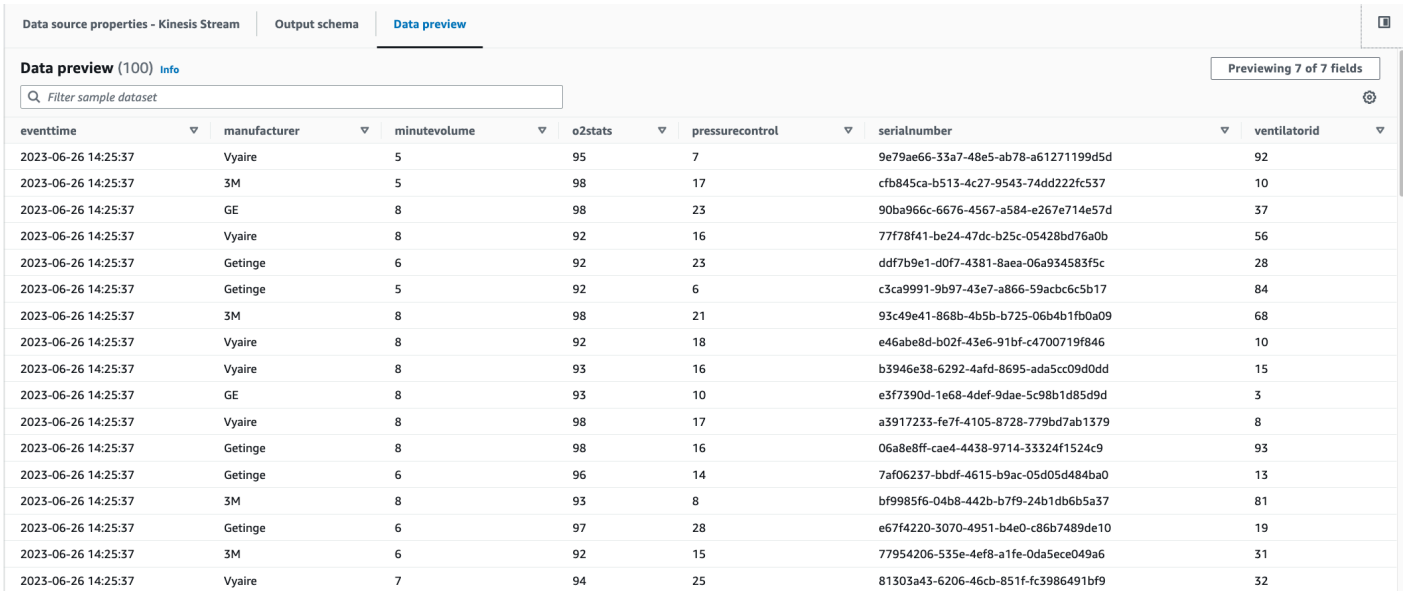
10. Seleccione Detalles del flujo para Origen de Amazon Kinesis en la pestaña Propiedades del origen de datos: flujo de Kinesis.
11. Seleccione El flujo se encuentra en mi cuenta para ver la Ubicación del flujo de datos.
12. Seleccione la región que está usando.
13. Seleccione el flujo de `GlueStreamTest-{AWS::AccountId}`.
14. Mantenga todas las otras opciones de configuración predeterminadas.



15. Navegue a la pestaña Vista previa de datos.
16. Haga clic en Iniciar sesión de vista previa de datos, la cual muestra una vista previa de los datos simulados generados por KDG. Elija el rol de servicio de Glue que creó anteriormente para el trabajo de transmisión de AWS Glue.

    Los datos de vista previa tardan entre 30 y 60 segundos en aparecer. Si aparece No hay datos para mostrar, haga clic en el icono con forma de engranaje y cambie el número de filas que desea muestrear a `100`.

    Puede ver los datos de muestra de la siguiente manera:



Data source properties - Kinesis Stream | Output schema | Data preview

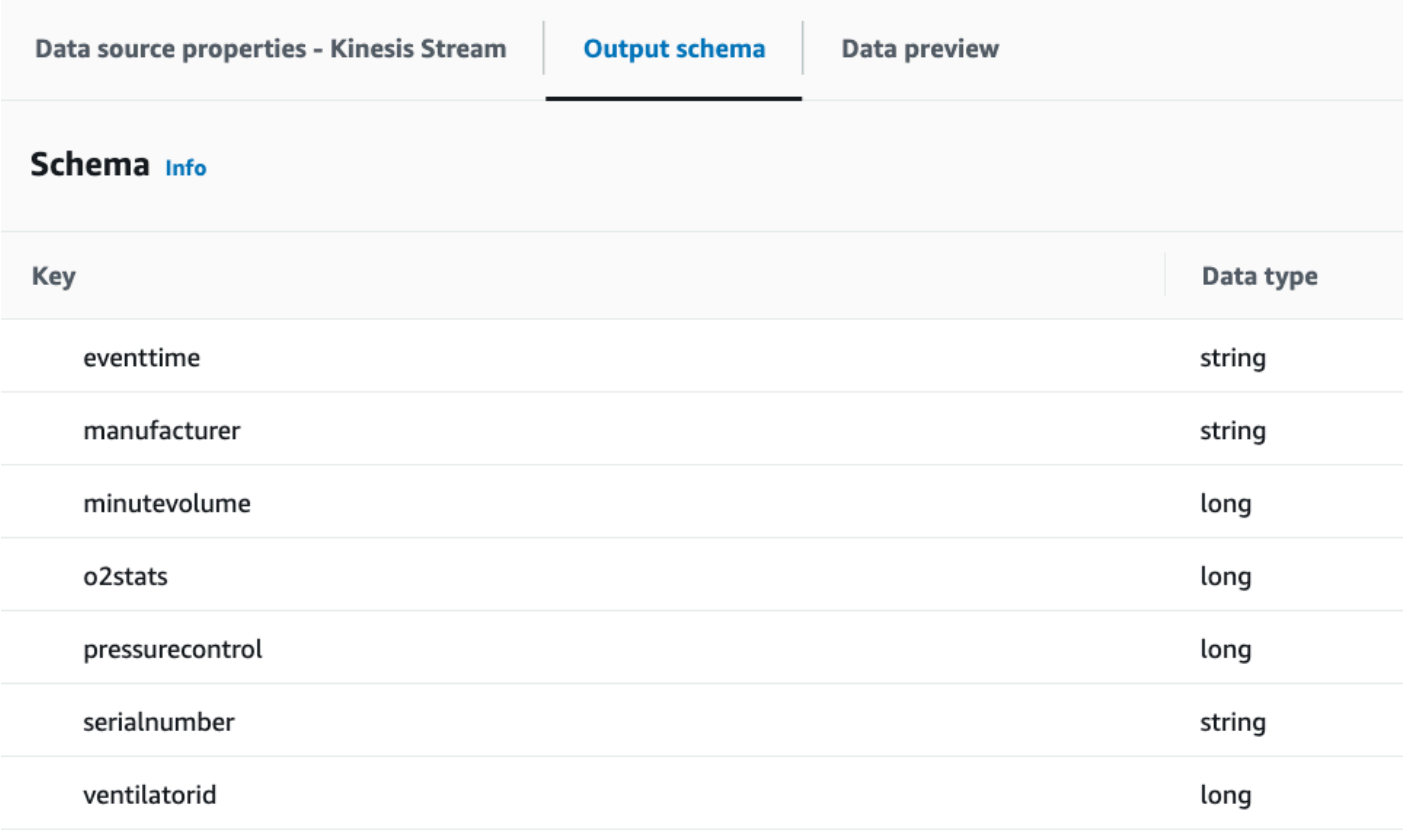
Data preview (100) Info

Previewing 7 of 7 fields

Filter sample dataset

eventtime	manufacturer	minutevolume	o2stats	pressurecontrol	serialnumber	ventilatorid
2023-06-26 14:25:37	Vyaire	5	95	7	9e79ae66-33a7-48e5-ab78-a61271199d5d	92
2023-06-26 14:25:37	3M	5	98	17	cfb845ca-b513-4c27-9543-74dd222fc537	10
2023-06-26 14:25:37	GE	8	98	23	90ba966c-6676-4567-a584-e267e714e57d	37
2023-06-26 14:25:37	Vyaire	8	92	16	77f78f41-be24-47dc-b25c-05428bd76a0b	56
2023-06-26 14:25:37	Getinge	6	92	23	ddf7b9e1-d0f7-4381-8aea-06a934583f5c	28
2023-06-26 14:25:37	Getinge	5	92	6	c3ca9991-9b97-43e7-a866-59acbc6c5b17	84
2023-06-26 14:25:37	3M	8	98	21	93c49e41-868b-4b5b-b725-06b4b1fb0a09	68
2023-06-26 14:25:37	Vyaire	8	92	18	e46abe8d-b02f-43e6-91bf-c4700719f846	10
2023-06-26 14:25:37	Vyaire	8	93	16	b3946e38-6292-4afd-8695-ada5cc09d0dd	15
2023-06-26 14:25:37	GE	8	93	10	e3f7390d-1e68-4def-9dae-5c98b1d85d9d	3
2023-06-26 14:25:37	Vyaire	8	98	17	a3917233-fe7f-4105-8728-779bd7ab1379	8
2023-06-26 14:25:37	Getinge	8	98	16	06a8e8ff-cae4-4438-9714-33324f1524c9	93
2023-06-26 14:25:37	Getinge	6	96	14	7af06237-bbdf-4615-b9ac-05d05d484ba0	13
2023-06-26 14:25:37	3M	8	93	8	bf9985f6-04b8-442b-b7f9-24b1db6b5a37	81
2023-06-26 14:25:37	Getinge	6	97	28	e67f4220-3070-4951-b4e0-c86b7489de10	19
2023-06-26 14:25:37	3M	6	92	15	77954206-535e-4ef8-a1fe-0da5ece049a6	31
2023-06-26 14:25:37	Vyaire	7	94	25	81303a43-6206-46cb-851f-fc3986491bf9	32

También puede ver el esquema inferido en la pestaña Esquema de salida.



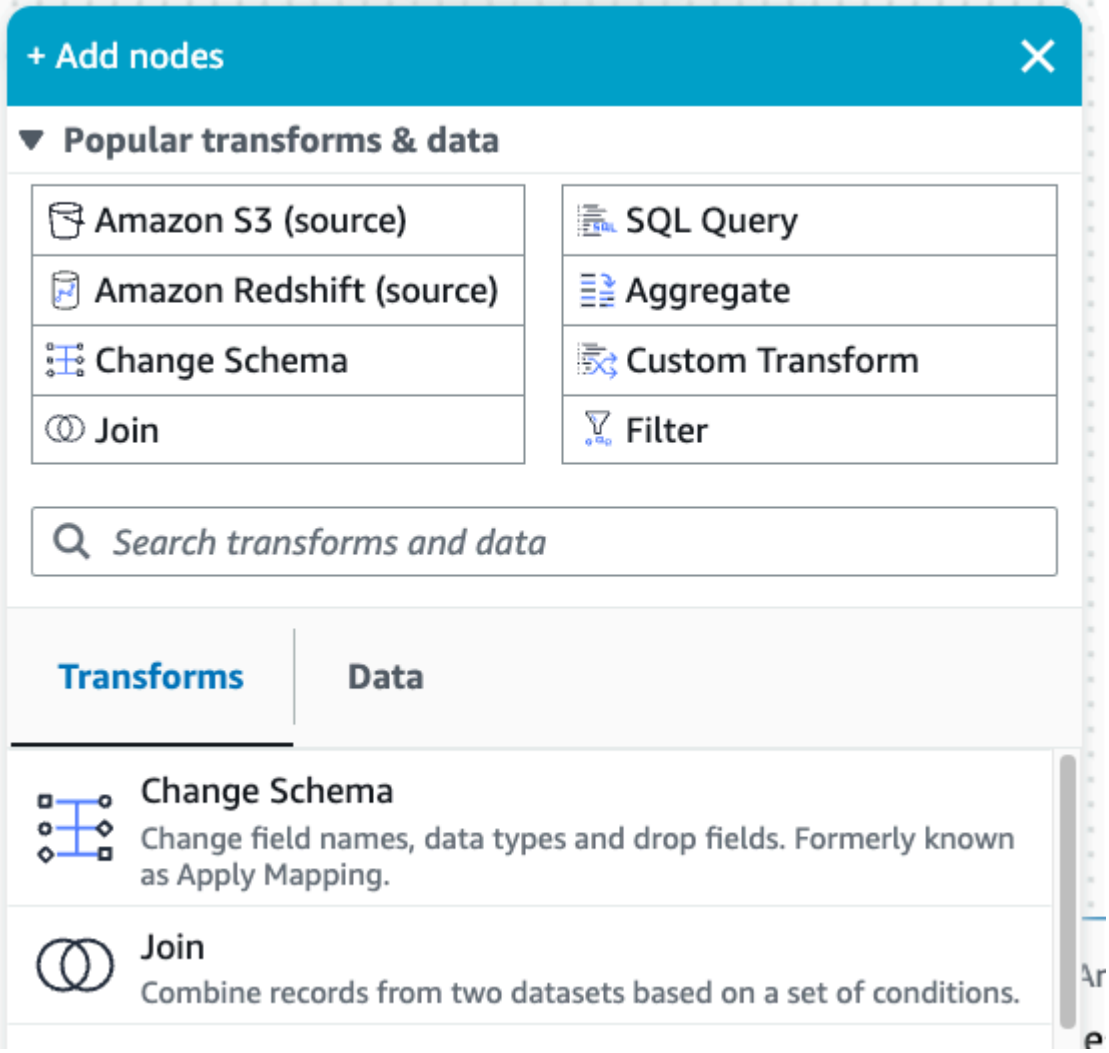
Data source properties - Kinesis Stream | Output schema | Data preview

Schema Info

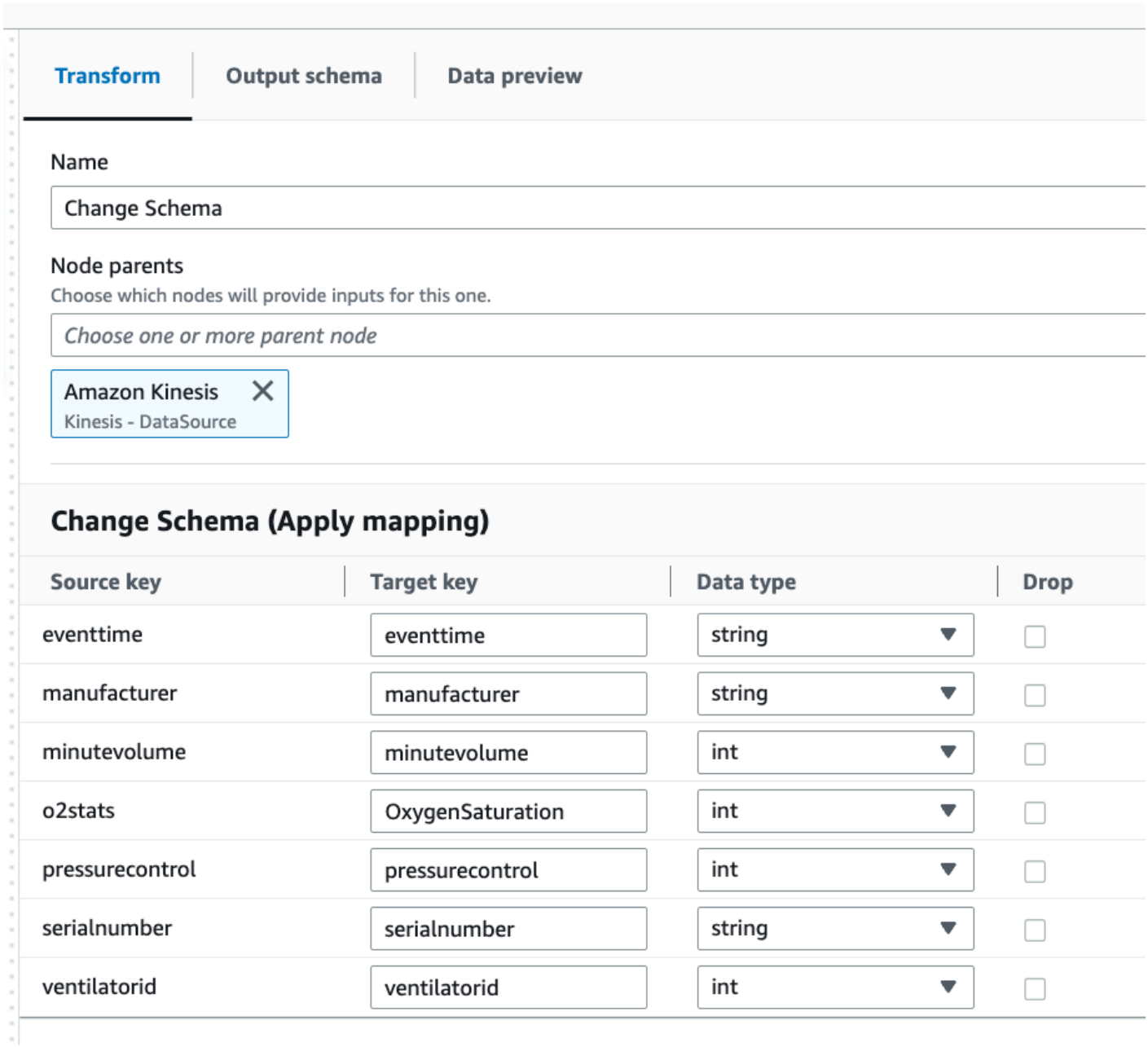
Key	Data type
eventtime	string
manufacturer	string
minutevolume	long
o2stats	long
pressurecontrol	long
serialnumber	string
ventilatorid	long

Transformación y almacenamiento del resultado transformado en Amazon S3

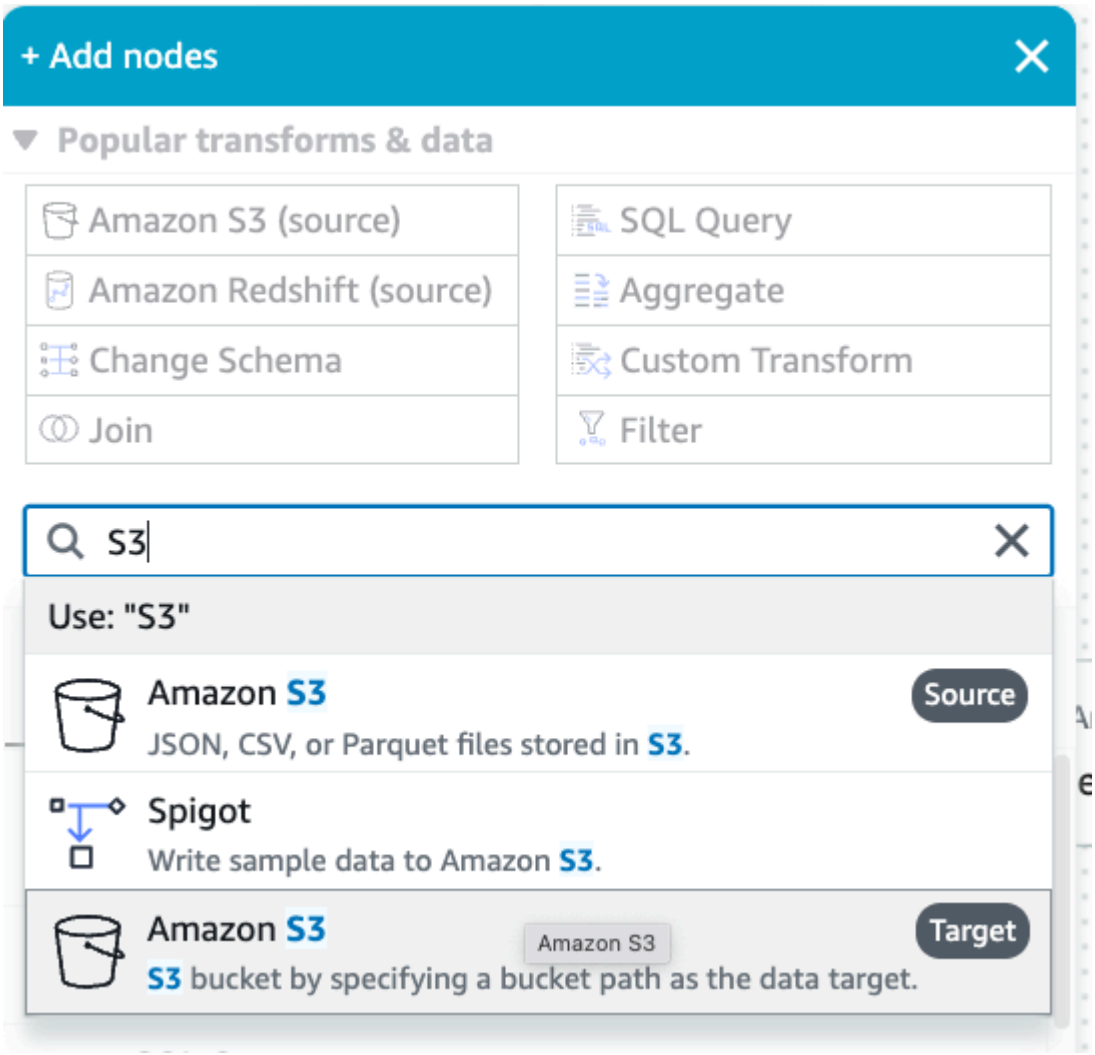
1. Con el nodo de origen seleccionado, haga clic en el icono con el signo más, situado en la parte superior izquierda, para agregar un paso de transformación.
2. Seleccione el paso Cambiar esquema.



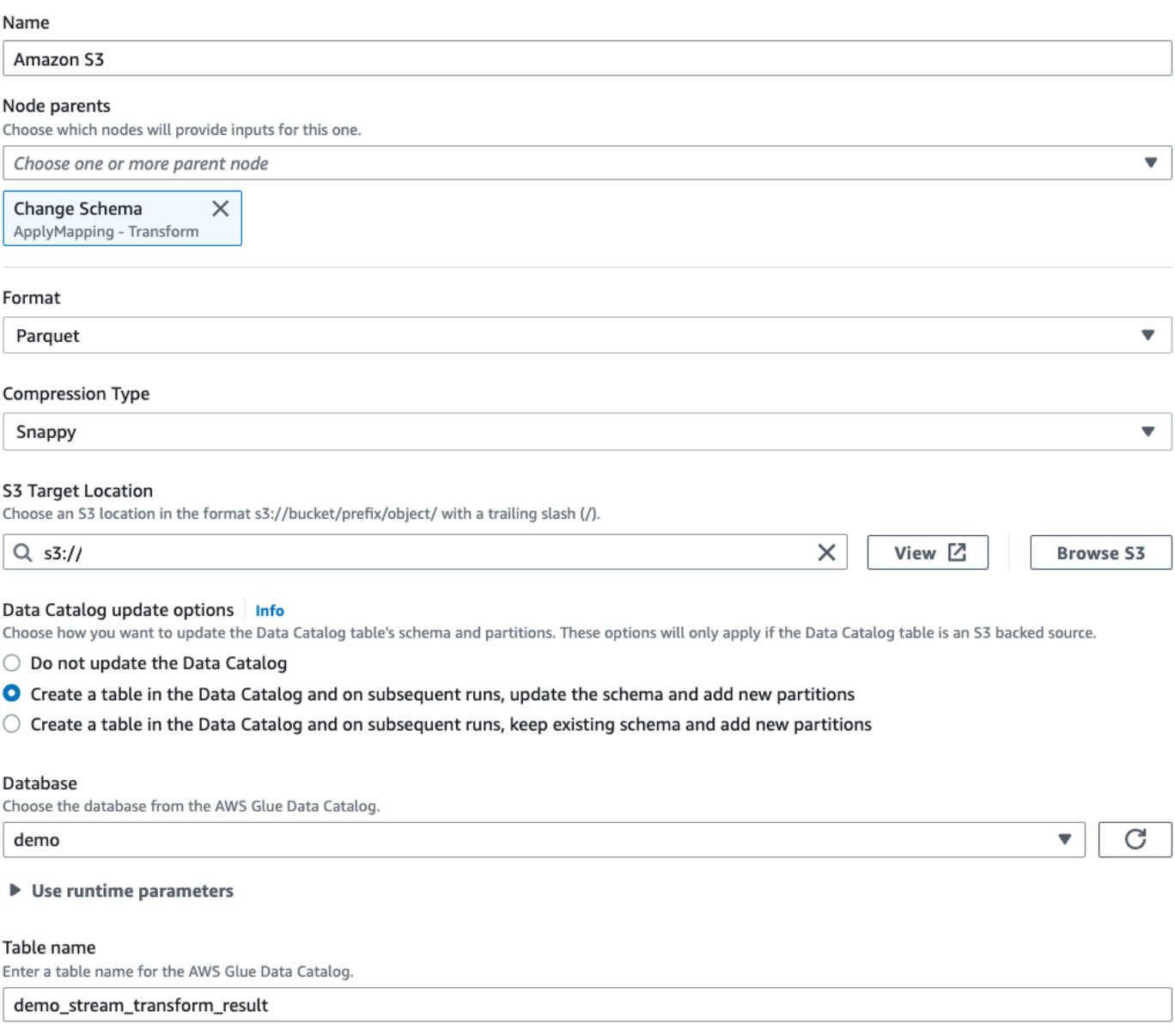
3. En este paso, puede cambiar el nombre de los campos y convertir el tipo de datos de los campos. Cambie el nombre de la columna `o2stats` a `OxygenSaturation` y convierta todos los tipos de datos de `long` a `int`.



4. Haga clic en el icono con el signo más para agregar un destino de Amazon S3. Introduzca S3 en el cuadro de búsqueda y seleccione el paso de transformación Amazon S3: destino.

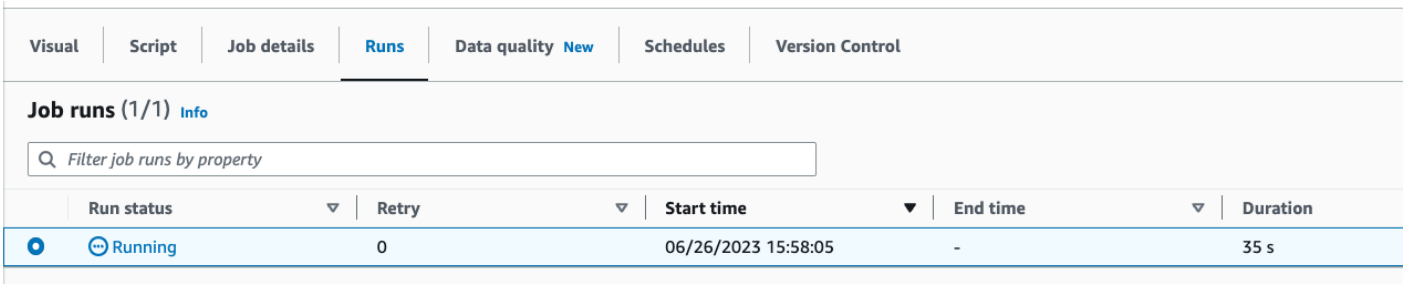


5. Seleccione Parquet como formato de archivo de destino.
6. Seleccione Snappy como tipo de compresión.
7. Introduzca una ubicación de destino de S3 creada por la plantilla de CloudFormation, `streaming-tutorial-s3-target-{AWS::AccountId}`.
8. Seleccione Crear una tabla en el Catálogo de datos y, en las ejecuciones posteriores, actualizar el esquema y agregar nuevas particiones.
9. Introduzca la base de datos de destino y el nombre de la tabla para almacenar el esquema de la tabla de destino de Amazon S3.

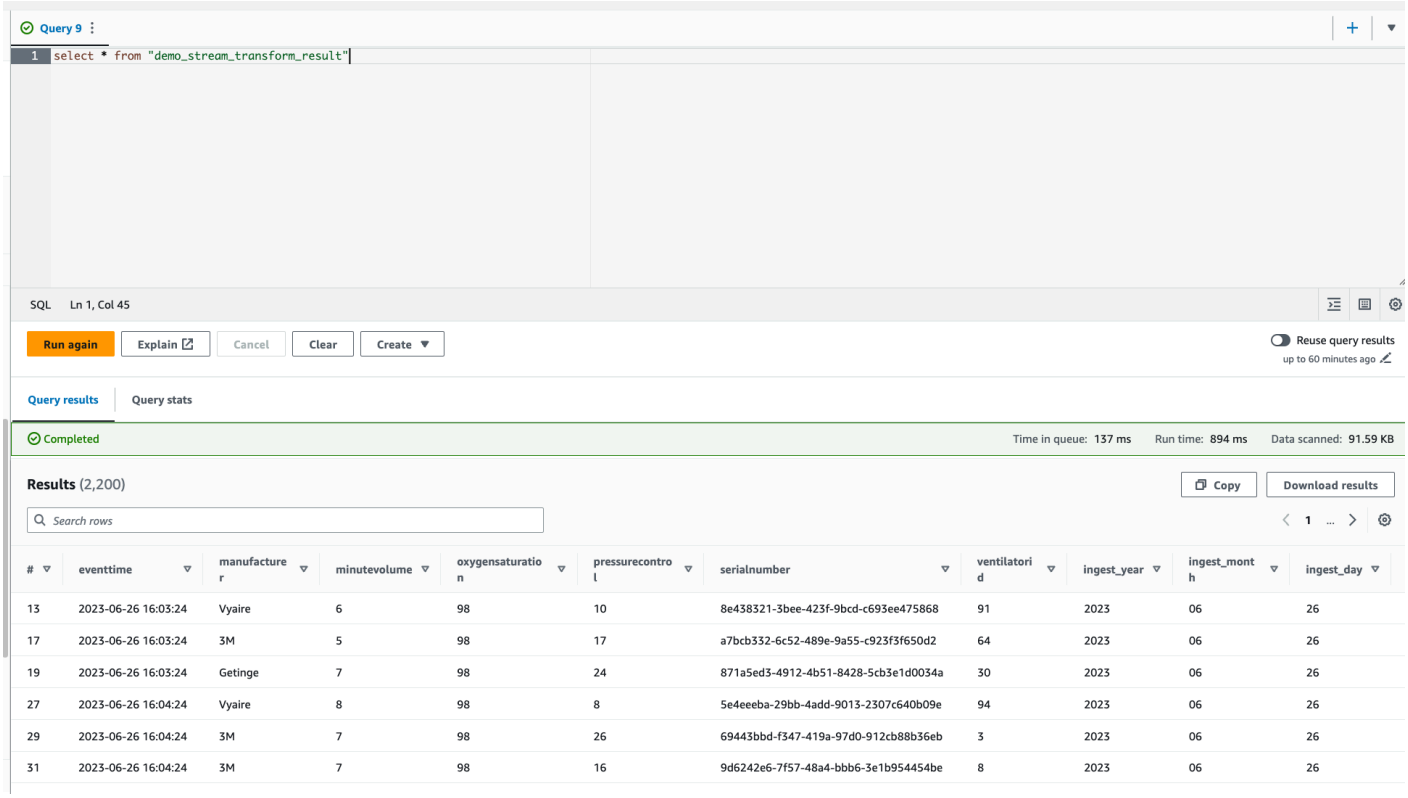


10. Haga clic en la pestaña Script para ver el código generado.
11. Haga clic en Guardar en la parte superior derecha para guardar el código ETL y, a continuación, haga clic en Ejecutar para iniciar el trabajo de transmisión de AWS Glue.

    Puede encontrar el estado de ejecución en la pestaña Ejecuciones. Deje que el trabajo se ejecute entre 3 y 5 minutos y, a continuación, deténgalo.



12. Compruebe la nueva tabla creada en Amazon Athena.



Tutorial: Cree su primera carga de trabajo de transmisión con los cuadernos de AWS Glue Studio

En este tutorial, explorará cómo aprovechar los cuadernos de AWS Glue Studio a fin de crear y refinar de forma interactiva sus trabajos de ETL para el procesamiento de datos casi en tiempo real. Si acaba de empezar a usar AWS Glue o si desea mejorar sus habilidades, esta guía le mostrará el proceso y le permitirá aprovechar todo el potencial de los cuadernos de las sesiones interactivas de AWS Glue.

Con la transmisión de AWS Glue, puede crear trabajos de extracción, transformación y carga (ETL) de transmisión que se ejecuten de forma continua y consuman datos de los orígenes de transmisión como Amazon Kinesis Data Streams, Apache Kafka y Amazon Managed Streaming para Apache Kafka (Amazon MSK).

Requisitos previos

Para seguir este tutorial, necesitará un usuario con permisos de consola de AWS para usar AWS Glue, Amazon Kinesis, Amazon S3, Amazon Athena, AWS CloudFormation, AWS Lambda y Amazon Cognito.

Consumir datos de transmisión desde Amazon Kinesis

Temas

- Generación de datos simulados con Kinesis Data Generator
- Creación de un trabajo de transmisión de AWS Glue con AWS Glue Studio
- Limpieza
- Conclusión

Generación de datos simulados con Kinesis Data Generator

> **Note**
>
> Si ya ha completado la sección anterior Tutorial: Cree su primera carga de trabajo de transmisión con AWS Glue Studio, ya tiene Kinesis Data Generator instalado en su cuenta, por lo que puede omitir los pasos 1 a 8 que aparecen a continuación y pasar a la sección Creación de un trabajo de transmisión de AWS Glue con AWS Glue Studio.

Puede generar sintéticamente datos de muestra en formato JSON mediante Kinesis Data Generator (KDG). Encontrará instrucciones y detalles completos en la documentación de la herramienta.

1. Para empezar, haga clic en

   para ejecutar una plantilla de AWS CloudFormation en su entorno de AWS.

> **Note**
>
> Es posible que se produzca un error en la plantilla de CloudFormation porque algunos recursos, como el usuario de Amazon Cognito para Kinesis Data Generator, ya existen en su cuenta de AWS. Esto puede deberse a que ya la configuró en otro tutorial o blog. Para solucionar este problema, puede probar la plantilla en una cuenta de AWS nueva para empezar de cero o explorar una región de AWS diferente. Estas opciones le permiten ejecutar el tutorial sin que entre en conflicto con los recursos existentes.

La plantilla aprovisiona un flujo de datos de Kinesis y una cuenta de Kinesis Data Generator.

2. Introduzca un nombre de usuario y una contraseña que KDG usará para autenticarse. Tome nota del nombre de usuario y la contraseña para su uso posterior.
3. Seleccione Siguiente hasta el último paso. Acepte la creación de recursos de IAM. Compruebe si hay algún error en la parte superior de la pantalla, como que la contraseña no cumple los requisitos mínimos, e implemente la plantilla.
4. Navegue hasta la pestaña Salidas de la pila. Una vez implementada la plantilla, mostrará la propiedad generada KinesisDataGeneratorUrl. Haga clic en esa URL.
5. Ingrese el nombre de usuario y la contraseña que anotó.
6. Seleccione la región que está usando y elija Kinesis Stream `GlueStreamTest-{AWS::AccountId}`.
7. Escriba la siguiente plantilla:

```
{
    "ventilatorid": {{random.number(100)}},
    "eventtime": "{{date.now("YYYY-MM-DD HH:mm:ss")}}",
    "serialnumber": "{{random.uuid}}",
    "pressurecontrol": {{random.number(
        {
            "min":5,
            "max":30
        }
    )}},
    "o2stats": {{random.number(
        {
            "min":92,
            "max":98
```

```
        }
    )}},
    "minutevolume": {{random.number(
        {
            "min":5,
            "max":8
        }
    )}},
    "manufacturer": "{{random.arrayElement(
        ["3M", "GE","Vyaire", "Getinge"]
    )}}"
}
```

Ahora puede ver los datos simulados con la plantilla de prueba e incorporarlos a Kinesis con Enviar datos.

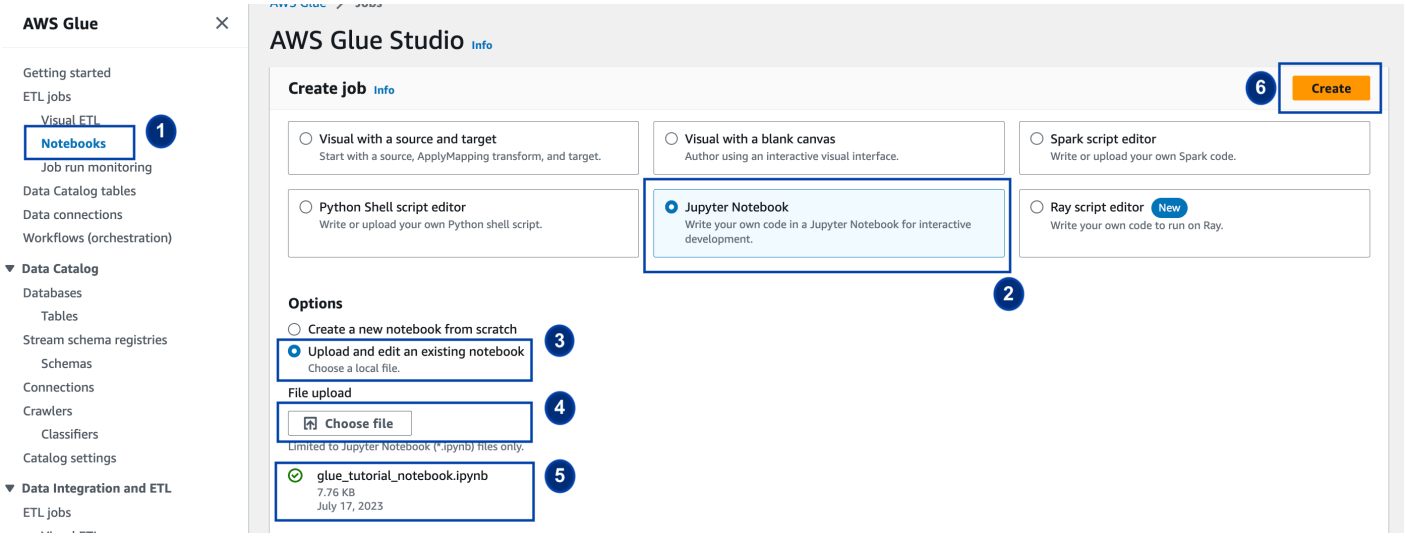
8. Haga clic en Enviar datos y genere entre 5 y 10 000 registros en Kinesis.

Creación de un trabajo de transmisión de AWS Glue con AWS Glue Studio

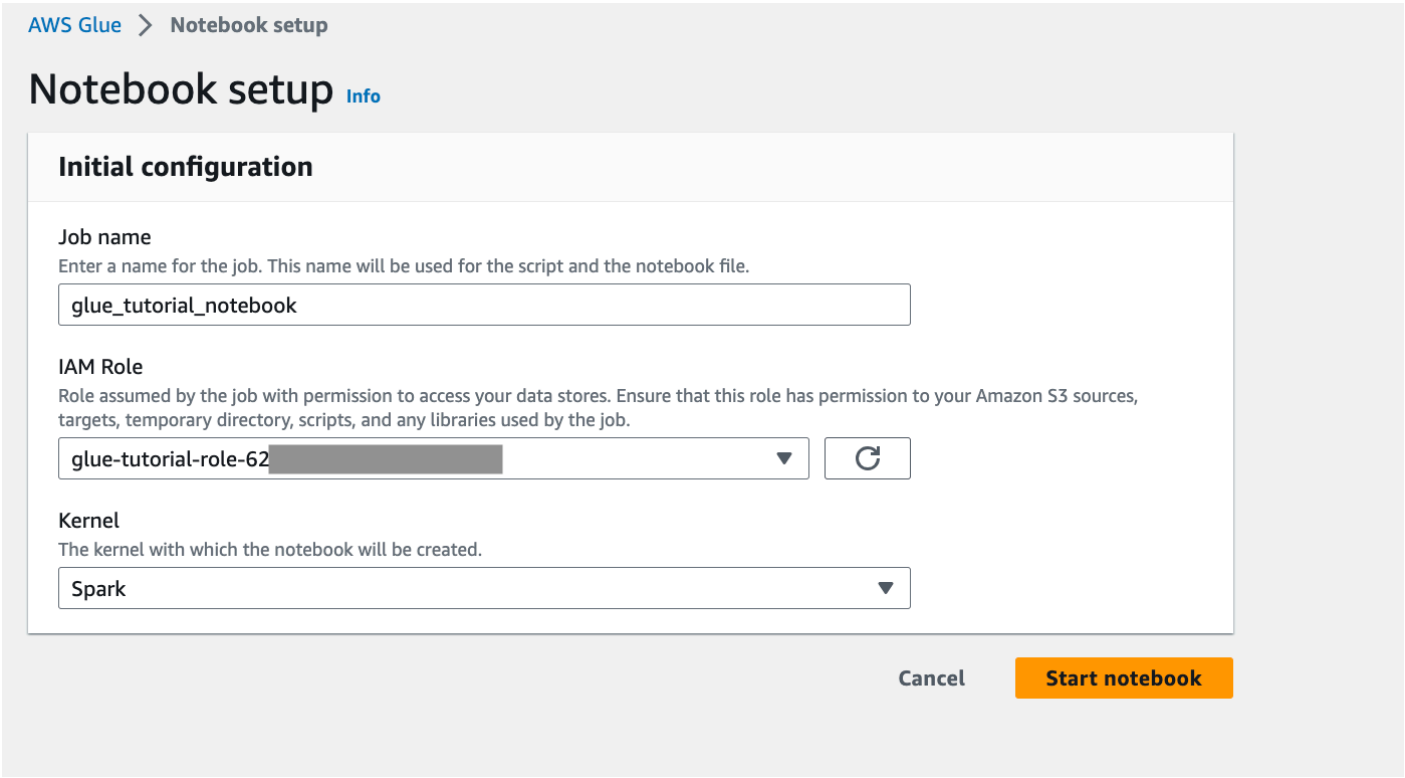
AWS Glue Studio es una interfaz visual que simplifica el proceso de diseño, organización y supervisión de las canalizaciones de integración de datos. Permite a los usuarios crear canalizaciones de transformación de datos sin necesidad de escribir un código extenso. Además de la experiencia visual de creación de trabajos, AWS Glue Studio también incluye un cuaderno de Jupyter respaldado por sesiones interactivas de AWS Glue, que usará en el resto de este tutorial.

Configure el trabajo de sesiones interactivas de la transmisión de AWS Glue

1. Descargue el archivo del cuaderno proporcionado y guárdelo en un directorio local.
2. Abra la consola de AWS Glue y, en el panel izquierdo, haga clic en Cuadernos > Cuaderno de Jupyter > Cargar y editar un cuaderno existente. Cargue el cuaderno del paso anterior y haga clic en Crear.



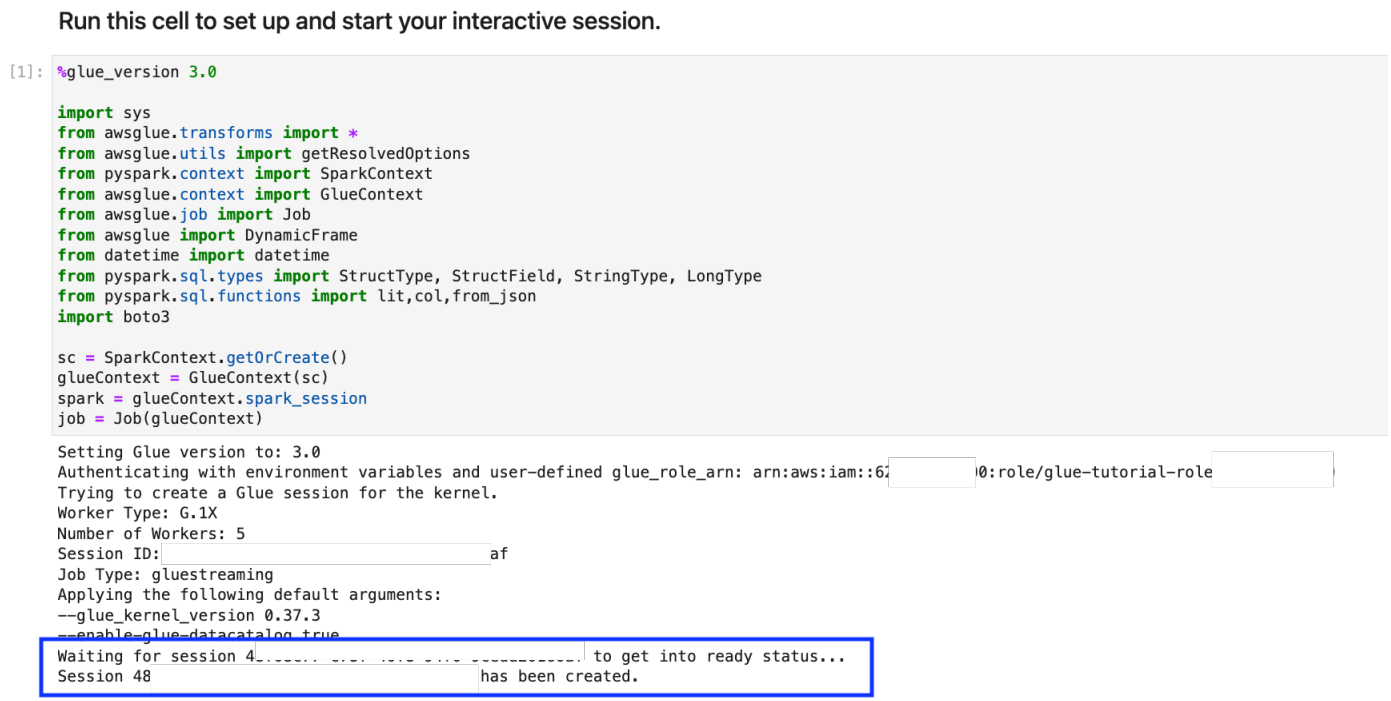
3. Proporcione un nombre y un rol al trabajo y seleccione el kernel de Spark predeterminado. A continuación, haga clic en Iniciar cuaderno. Para el rol de IAM, seleccione el rol aprovisionado en la plantilla de CloudFormation. Puede verlo en la pestaña Salidas de CloudFormation.



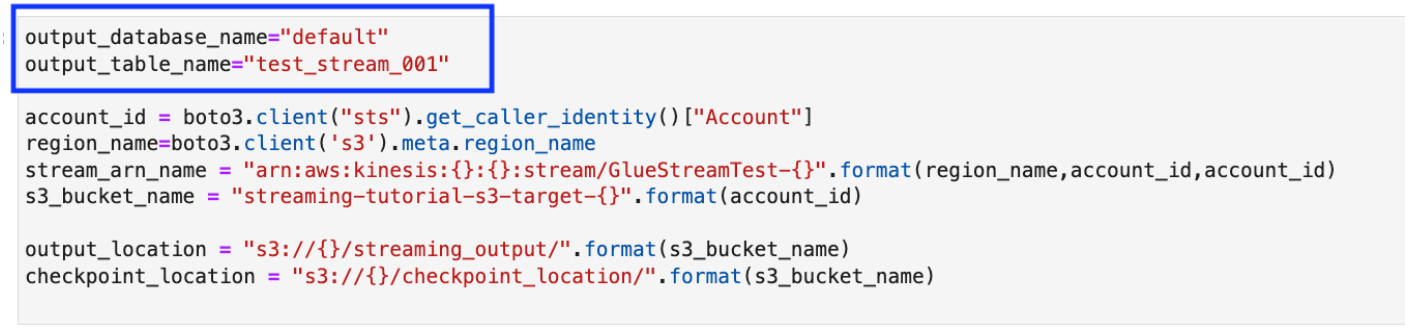
El cuaderno tiene todas las instrucciones necesarias para continuar con el tutorial. Puede ejecutar las instrucciones en el cuaderno o seguir este tutorial para continuar con el desarrollo del trabajo.

Ejecute las celdas del cuaderno

1. (Opcional) La primera celda de código `%help` muestra todos los comandos mágicos disponibles del cuaderno. Puede saltarse esta celda por ahora, pero no dude en explorarla.
2. Empiece con el siguiente bloque de código `%streaming`. Este comando mágico define el tipo de trabajo como transmisión, lo que le permite desarrollar, depurar e implementar un trabajo de ETL de transmisión de AWS Glue.
3. Ejecute la siguiente celda para crear una sesión interactiva de AWS Glue. La celda de salida tiene un mensaje que confirma la creación de la sesión.



4. La siguiente celda define las variables. Sustituya los valores por otros adecuados para su trabajo y ejecute la celda. Por ejemplo:



5. Como los datos ya se están transmitiendo a Kinesis Data Streams, la siguiente celda consumirá los resultados del flujo. Ejecute la siguiente celda. Como no hay instrucciones de impresión, no se espera ningún resultado de esta celda.

6. En la siguiente celda, se explora el flujo entrante tomando un conjunto de muestra e imprimiendo su esquema y los datos reales. Por ejemplo:

Sample and print the incoming records

the sampling is for debugging purpose. You may comment off the entire code cell below, before deploying the actual code

[4]:
```
options = {
    "pollingTimeInMs": "20000",
    "windowSize": "5 seconds"
}
sampled_dynamic_frame = glueContext.getSampleStreamingDynamicFrame(data_frame, options, None)

count_of_sampled_records = sampled_dynamic_frame.count()

print(count_of_sampled_records)

sampled_dynamic_frame.printSchema()

sampled_dynamic_frame.toDF().show(10,False)
```

```
100
root
|-- eventtime: string
|-- manufacturer: string
|-- minutevolume: long
|-- o2stats: long
|-- pressurecontrol: long
|-- serialnumber: string
|-- ventilatorid: long

+-------------------+------------+------------+-------+---------------+------------------------------------+------------+
|eventtime          |manufacturer|minutevolume|o2stats|pressurecontrol|serialnumber                        |ventilatorid|
+-------------------+------------+------------+-------+---------------+------------------------------------+------------+
|2023-07-18 10:20:11|3M          |6           |92     |24             |a3e860ba-24b9-41c4-bc10-91c6b35e1406|6           |
|2023-07-18 10:20:11|Vyaire      |6           |95     |6              |96101dca-3e88-457f-b390-e3291df48a81|26          |
|2023-07-18 10:20:12|Getinge     |8           |96     |24             |18f3d448-1dee-4c80-835b-1a0daa818915|22          |
|2023-07-18 10:20:12|Getinge     |7           |98     |30             |25f425cd-b978-4953-9a03-4d607a639364|91          |
|2023-07-18 10:20:12|GE          |5           |93     |25             |2cd7cdc2-f5f5-4ff2-ae32-45e5a8922d53|93          |
```

7. A continuación, defina la lógica de transformación de datos real. La celda consiste en el método `processBatch` que se activa durante cada microlote. Ejecute la celda. En un nivel superior, haga lo siguiente con el flujo entrante:

   a. Seleccione un subconjunto de las columnas de entrada.

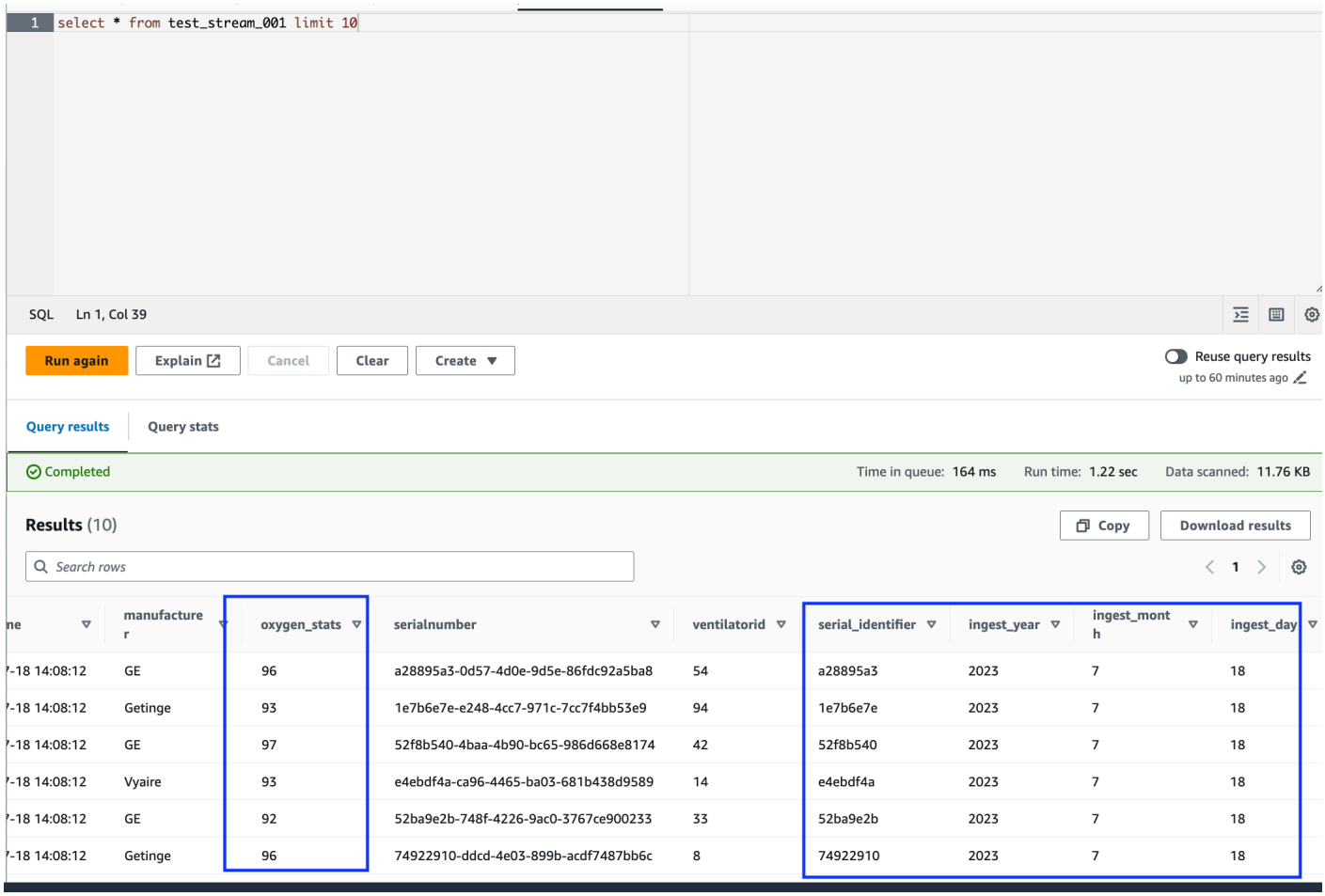
   b. Cambie el nombre de una columna (o2stats a oxygen_stats).

   c. Derive columnas nuevas (serial_identifier, ingest_year, ingest_month e ingest_day).

   d. Almacene los resultados en un bucket de Amazon S3 y también cree una tabla de catálogo de AWS Glue particionada.

8. En la última celda, active el proceso por lotes cada 10 segundos. Ejecute la celda y espere unos 30 segundos para que llene el bucket de Amazon S3 y la tabla del catálogo de AWS Glue.

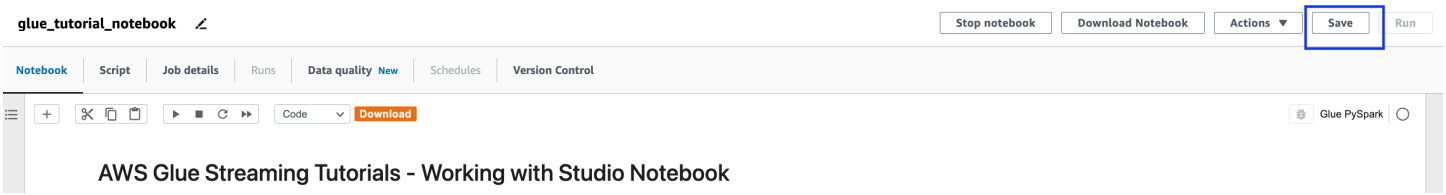
9. Por último, explore los datos almacenados con el editor de consultas de Amazon Athena. Puede ver la columna a la que se le ha cambiado el nombre y también las nuevas particiones.



El cuaderno tiene todas las instrucciones necesarias para continuar con el tutorial. Puede ejecutar las instrucciones en el cuaderno o seguir este tutorial para continuar con el desarrollo del trabajo.

Guarde y ejecute el trabajo de AWS Glue

Una vez que haya completado el desarrollo y las pruebas de su aplicación con el cuaderno de sesiones interactivas, haga clic en Guardar en la parte superior de la interfaz del cuaderno. Una vez guardado, también puede ejecutar la aplicación como un trabajo.



Limpieza

Para evitar incurrir en cargos adicionales en su cuenta, detenga el trabajo de transmisión que inició siguiendo las instrucciones. Puede hacerlo cerrando el cuaderno, lo que finalizará la sesión. Vacíe el bucket de Amazon S3 y elimine la pila de AWS CloudFormation que aprovisionó anteriormente.

Conclusión

En este tutorial, mostramos cómo usar el cuaderno de AWS Glue Studio para hacer lo siguiente:

- Crear un trabajo de ETL de transmisión con cuadernos
- Previsualizar los flujos de datos entrantes
- Codificar y solucionar problemas sin tener que publicar los trabajos de AWS Glue
- Revisar el código de trabajo de principio a fin, eliminar cualquier error e imprimir las instrucciones o celdas del cuaderno
- Publicar el código como un trabajo de AWS Glue

El objetivo de este tutorial es proporcionarle experiencia práctica al trabajar con la transmisión de AWS Glue y las sesiones interactivas. Le recomendamos que lo use como referencia para sus casos de uso individuales de transmisión de AWS Glue. Para obtener más información, consulte Introducción a las sesiones interactivas de AWS Glue.

Conceptos de la transmisión de AWS Glue

En las siguientes secciones, se ofrece información acerca del los conceptos de la transmisión de AWS Glue.

Temas

- Anatomía de un trabajo de transmisión de AWS Glue
- Conexiones de Kafka
- Conexión de Kinesis
- AWS Glue Opciones de streaming

Anatomía de un trabajo de transmisión de AWS Glue

Los trabajos de transmisión de AWS Glue funcionan según el paradigma de transmisión de Spark y aprovechan la transmisión estructurada del marco de trabajo de Spark. Los trabajos de transmisión sondean constantemente el origen de datos de transmisión, en un intervalo específico, para obtener los registros en forma de microlotes. En las siguientes secciones, se examinan las distintas partes de un trabajo de transmisión de AWS Glue.


```
def processBatch(data_frame, batchId):
    if data_frame.count() > 0:
        AmazonKinesis_node1696872487972 = DynamicFrame.fromDF(
            glueContext.add_ingestion_time_columns(data_frame, "hour"),
            glueContext,
            "from_data_frame",
        )
        # Script generated for node Change Schema
        ChangeSchema_node1696872679326 = ApplyMapping.apply(
            frame=AmazonKinesis_node1696872487972,
            mappings=[
                ("eventtime", "string", "eventtime", "string"),
                ("manufacturer", "string", "manufacturer", "string"),
                ("minutevolume", "long", "minutevolume", "int"),
                ("o2stats", "long", "OxygenSaturation", "int"),
                ("pressurecontrol", "long", "pressurecontrol", "int"),
                ("serialnumber", "string", "serialnumber", "string"),
                ("ventilatorid", "long", "ventilatorid", "long"),
                ("ingest_year", "string", "ingest_year", "string"),
                ("ingest_month", "string", "ingest_month", "string"),
                ("ingest_day", "string", "ingest_day", "string"),
                ("ingest_hour", "string", "ingest_hour", "string"),
            ],
            transformation_ctx="ChangeSchema_node1696872679326",
        )

        # Script generated for node Amazon S3
        AmazonS3_node1696872743449_path = (
            "s3://streaming-tutorial-s3-target-[redacted]"
        )
        AmazonS3_node1696872743449 = glueContext.getSink(
            path=AmazonS3_node1696872743449_path,
            connection_type="s3",
            updateBehavior="UPDATE_IN_DATABASE",
            partitionKeys=["ingest_year", "ingest_month", "ingest_day", "ingest_hour"],
            compression="snappy",
            enableUpdateCatalog=True,
            transformation_ctx="AmazonS3_node1696872743449",
        )
        AmazonS3_node1696872743449.setCatalogInfo(
            catalogDatabase="demo", catalogTableName="demo_stream_trasform_result"
        )
        AmazonS3_node1696872743449.setFormat("glueparquet")
        AmazonS3_node1696872743449.writeFrame(ChangeSchema_node1696872679326)

glueContext.forEachBatch(
    frame=dataframe_AmazonKinesis_node1696872487972,
    batch_function=processBatch,
    options={
        "windowSize": "100 seconds",
        "checkpointLocation": args["TempDir"] + "/" + args["JOB_NAME"] + "/checkpoint/",
    },
)
job.commit()
```

forEachBatch

El método `forEachBatch` es el punto de entrada de la ejecución de un trabajo de transmisión de AWS Glue. Los trabajos de transmisión de AWS Glue usan el método `forEachBatch` para sondear los datos, lo cual funciona como un iterador que permanece activo durante la duración del trabajo de transmisión y sondea periódicamente el origen de transmisión en busca de nuevos datos y procesa los datos más recientes en microlotes.

```
glueContext.forEachBatch(
    frame=dataFrame_AmazonKinesis_node1696872487972,
    batch_function=processBatch,
```

```
    options={
        "windowSize": "100 seconds",
        "checkpointLocation": args["TempDir"] + "/" + args["JOB_NAME"] + "/
checkpoint/",
    },
)
```

Configure la propiedad `frame` de `forEachBatch` para especificar un origen de transmisión. En este ejemplo, el nodo de origen que creó en el lienzo en blanco durante la creación del trabajo se rellena con el DataFrame predeterminado del trabajo. Defina la propiedad `batch_function` como la `function` que decida invocar para cada operación de microlotes. Debe definir una función para gestionar la transformación por lotes de los datos entrantes.

Origen

En el primer paso de la función `processBatch`, el programa verifica el recuento de registros del DataFrame que usted definió como propiedad del marco de `forEachBatch`. El programa agrega una marca de tiempo de ingesta a un DataFrame que no esté vacío. La cláusula `data_frame.count()>0` determina si el último microlote no está vacío y está listo para su posterior procesamiento.

```
def processBatch(data_frame, batchId):
    if data_frame.count() >0:
        AmazonKinesis_node1696872487972 = DynamicFrame.fromDF(
            glueContext.add_ingestion_time_columns(data_frame, "hour"),
            glueContext,
            "from_data_frame",
        )
```

Correspondencia

La siguiente sección del programa consiste en aplicar la asignación. El método `Mapping.apply` de un DataFrame de Spark le permite definir una regla de transformación en torno a los elementos de datos. Por lo general, puede cambiar el nombre y el tipo de datos o aplicar una función personalizada en la columna de datos de origen y asignarla a las columnas de destino.

```
    #Script generated for node ChangeSchema
    ChangeSchema_node16986872679326 = ApplyMapping.apply(
```

```
        frame =  AmazonKinesis_node1696872487972,
        mappings = [
            ("eventtime", "string", "eventtime", "string"),
            ("manufacturer", "string", "manufacturer", "string"),
            ("minutevolume", "long", "minutevolume", "int"),
            ("o2stats", "long", "OxygenSaturation", "int"),
            ("pressurecontrol", "long", "pressurecontrol", "int"),
            ("serialnumber", "string", "serialnumber", "string"),
            ("ventilatorid", "long", "ventilatorid", "long"),
            ("ingest_year", "string", "ingest_year", "string"),
            ("ingest_month", "string", "ingest_month", "string"),
            ("ingest_day", "string", "ingest_day", "string"),
            ("ingest_hour", "string", "ingest_hour", "string"),
        ],
        transformation_ctx="ChangeSchema_node16986872679326",
    )
        )
```

Receptor

En esta sección, el conjunto de datos entrantes del origen de transmisión se almacena en una ubicación de destino. En este ejemplo, escribiremos los datos en una ubicación de Amazon S3. Los detalles de la propiedad `AmazonS3_node_path` se rellenan automáticamente según lo determinado por la configuración que usó durante la creación del trabajo desde el lienzo. Puede configurar el `updateBehavior` en función de su caso de uso y decidir no actualizar la tabla del catálogo de datos o crear un catálogo de datos y actualizar el esquema del catálogo de datos en las ejecuciones posteriores, o bien crear una tabla de catálogo y no actualizar la definición del esquema en las ejecuciones posteriores.

La propiedad `partitionKeys` define la opción de partición de almacenamiento. El comportamiento predeterminado es particionar los datos según el `ingestion_time_columns` que estaba disponible en la sección de origen. La propiedad `compression` permite configurar el algoritmo de compresión que se aplicará durante la escritura de destino. Tiene la opción de configurar Snappy, LZO o GZIP como la técnica de compresión. La propiedad `enableUpdateCatalog` controla si es necesario actualizar la tabla del catálogo de AWS Glue. Las opciones disponibles para esta propiedad son `True` o `False`.

```
    #Script generated for node Amazon S3
    AmazonS3_node1696872743449 = glueContext.getSink(
```

```
        path =  AmazonS3_node1696872743449_path,
        connection_type = "s3",
        updateBehavior = "UPDATE_IN_DATABASE",
        partitionKeys = ["ingest_year", "ingest_month", "ingest_day", "ingest_hour"],
        compression = "snappy",
        enableUpdateCatalog = True,
        transformation_ctx = "AmazonS3_node1696872743449",
    )
```

Receptor del catálogo de AWS Glue

Esta sección del trabajo controla el comportamiento de actualización de la tabla del catálogo de AWS Glue. Defina las propiedades `catalogDatabase` y `catalogTableName` según el nombre de la base de datos de su catálogo de AWS Glue y el nombre de la tabla asociada al trabajo de AWS Glue que está diseñando. Puede definir el formato de archivo de los datos de destino mediante la propiedad `setFormat`. Para este ejemplo, almacenaremos los datos en formato parquet.

Una vez que haya configurado y ejecutado el trabajo de transmisión de AWS Glue siguiendo este tutorial, los datos de transmisión producidos en Amazon Kinesis Data Streams se almacenarán en la ubicación de Amazon S3 en formato parquet con una compresión rápida. Si el trabajo de transmisión se ejecuta correctamente, podrá consultar los datos a través de Amazon Athena.

```
    AmazonS3_node1696872743449 = setCatalogInfo(
        catalogDatabase =  "demo", catalogTableName = "demo_stream_transform_result"
    )
    AmazonS3_node1696872743449.setFormat("glueparquet")
    AmazonS3_node1696872743449.writeFormat("ChangeSchema_node16986872679326")
    )
```

Conexiones de Kafka

Designa una conexión a un clúster de Kafka o a un clúster de Amazon Managed Streaming for Apache Kafka.

Puede leer y escribir en los flujos de datos de Kafka con la información almacenada en la tabla del Catálogo de datos o al brindar la información para acceder de manera directa al flujo de datos.

Puedes leer información de Kafka en una Spark DataFrame y luego convertirla en una AWS Glue DynamicFrame. Puedes escribir DynamicFrames en Kafka en formato JSON. Si accede directamente a la secuencia de datos, utilice estas opciones para proporcionar información sobre cómo acceder a la secuencia de datos.

Si utiliza `getCatalogSource` o `create_data_frame_from_catalog` para consumir los registros de un origen de streaming de Kafka, o `getCatalogSink` o `write_dynamic_frame_from_catalog` para escribir registros en Kafka, el trabajo cuenta con la base de datos del Catálogo de datos y la información del nombre de la tabla, lo cual se puede utilizar para obtener algunos parámetros básicos para la lectura de un origen de streaming de Kafka. Si utiliza `getSource`, `getCatalogSink`, `getSourceWithFormat`, `getSinkWithFormat`, `createDataFrameFromOptions`, `create_data_frame_from_options` o `write_dynamic_frame_from_catalog`, debe especificar estos parámetros básicos con las opciones de conexión que se describen aquí.

Puede especificar las opciones de conexión para Kafka con los argumentos que se mencionan a continuación para los métodos especificados en la clase `GlueContext`.

- Scala
  - `connectionOptions`: se debe utilizar con `getSource`, `createDataFrameFromOptions` y `getSink`
  - `additionalOptions`: se debe utilizar con `getCatalogSource`, `getCatalogSink`
  - `options`: se debe utilizar con `getSourceWithFormat`, `getSinkWithFormat`
- Python
  - `connection_options`: se debe utilizar con `create_data_frame_from_options`, `write_dynamic_frame_from_options`
  - `additional_options`: se debe utilizar con `create_data_frame_from_catalog`, `write_dynamic_frame_from_catalog`
  - `options`: se debe utilizar con `getSource`, `getSink`

Para obtener notas y conocer las restricciones sobre los trabajos de ETL de transmisión, consulte the section called “Notas y restricciones de ETL de streaming”.

Configurar Kafka

No hay AWS requisitos previos para conectarse a las transmisiones de Kafka disponibles a través de Internet.

Puedes crear una conexión AWS Glue Kafka para gestionar tus credenciales de conexión. Para obtener más información, consulte the section called “Crear una conexión para un flujo de datos Kafka”. En la configuración del trabajo de AWS Glue, *introduce ConnectionName* como conexión de red adicional y, a continuación, en la llamada al método, introduce *ConnectionName* al parámetro. `connectionName`

En algunos casos, tendrá que configurar requisitos previos adicionales:

- Si utiliza Amazon Managed Streaming for Apache Kafka con autenticación de IAM, necesitará una configuración de IAM adecuada.
- Si utiliza Amazon Managed Streaming for Apache Kafka con una VPC de Amazon, necesitará una configuración de VPC de Amazon adecuada. Deberás crear una conexión AWS Glue que proporcione la información de conexión de Amazon VPC. Necesitará que la configuración de su trabajo incluya la conexión AWS Glue como conexión de red adicional.

Para obtener más información sobre los requisitos previos del trabajo de ETL de Streaming, consulte the section called “Trabajos ETL de streaming”.

Ejemplo: leer desde transmisiones de Kafka

Se utiliza junto con the section called “forEachBatch”.

Ejemplo de origen de streaming de Kafka:

```
kafka_options =
    { "connectionName": "ConfluentKafka",
      "topicName": "kafka-auth-topic",
      "startingOffsets": "earliest",
      "inferSchema": "true",
      "classification": "json"
    }
data_frame_datasource0 =
 glueContext.create_data_frame.from_options(connection_type="kafka",
 connection_options=kafka_options)
```

Ejemplo: escritura a flujos de Kafka

Ejemplos de escritura a Kafka:

Ejemplo con el método `getSink`:

```
data_frame_datasource0 =
glueContext.getSink(
 connectionType="kafka",
 connectionOptions={
  JsonOptions("""{
   "connectionName": "ConfluentKafka",
   "classification": "json",
   "topic": "kafka-auth-topic",
   "typeOfData": "kafka"}
  """)},
 transformationContext="dataframe_ApacheKafka_node1711729173428")
 .getDataFrame()
```

Ejemplo con el método `write_dynamic_frame.from_options`:

```
kafka_options =
    { "connectionName": "ConfluentKafka",
      "topicName": "kafka-auth-topic",
      "classification": "json"
    }
data_frame_datasource0 =
 glueContext.write_dynamic_frame.from_options(connection_type="kafka",
 connection_options=kafka_options)
```

Referencia de opciones de conexión de Kafka

Al leer, utilice las opciones de conexión con `"connectionType": "kafka"` a continuación:

- `"bootstrap.servers"` (obligatorio): una lista de direcciones URL Bootstrap, por ejemplo, como `b-1.vpc-test-2.o4q88o.c6.kafka.us-east-1.amazonaws.com:9094`. Esta opción debe especificarse en la llamada a la API o definirse en los metadatos de la tabla en el Data Catalog.
- `"security.protocol"` (obligatorio): el protocolo que se utiliza para la comunicación con los agentes. Los valores posibles son `"SSL"` o.`"PLAINTEXT"`
- `"topicName"` (Obligatorio) Lista separada por comas de temas a los que suscribirse. Debe especificar solo una opción de `"topicName"`, `"assign"` o `"subscribePattern"`.
- `"assign"`: (Obligatorio) Una cadena JSON que especifica el valor de `TopicPartitions` para consumir. Debe especificar solo una opción de `"topicName"`, `"assign"` o `"subscribePattern"`.

  Ejemplo: “{"temaA":[0,1],"temaB":[2,4]}”

- `"subscribePattern"`: (obligatorio) cadena de expresiones regulares de Java que identifica la lista de temas a la que desea suscribirse. Debe especificar solo una opción de `"topicName"`, `"assign"` o `"subscribePattern"`.

  Ejemplo: “tema.*”

- `"classification"` (Obligatorio) El formato de archivo utilizado por los datos del registro. Obligatorio a menos que se proporcione a través del catálogo de datos.
- `"delimiter"` (Opcional) El separador de valores que se utiliza cuando `classification` es CSV. El valor predeterminado es “`,`”.
- `"startingOffsets"`: (opcional) posición inicial en el tema de Kafka para leer los datos. Los valores posibles son `"earliest"` o.`"latest"` El valor predeterminado es `"latest"`.
- `"startingTimestamp"`: (Opcional, solo compatible con la versión 4.0 o posterior de AWS Glue) La marca de tiempo del registro del tema de Kafka del que se van a leer los datos. Los valores posibles son una cadena de marca de tiempo en formato UTC en el patrón `yyyy-mm-ddTHH:MM:SSZ` (donde `Z` representa un desplazamiento de zona horaria UTC con un +/-. Por ejemplo, “2023-04-04T08:00:00-04:00”).

  Nota: Solo una de las opciones «StartingOffsets» o «StartingTimestamp» puede estar presente en la lista de opciones de conexión del script de streaming de AWS Glue. Si se incluyen estas dos propiedades, se producirá un error en el trabajo.

- `"endingOffsets"`: (opcional) el punto final cuando finaliza una consulta por lotes. Los valores posibles son `"latest"` o una cadena JSON que especifica una compensación final para cada `TopicPartition`.

  Para la cadena JSON, el formato es `{"topicA":{"0":23,"1":-1},"topicB":{"0":-1}}`. El valor `-1` como compensación representa `"latest"`.

- `"pollTimeoutMs"`: (opcional) tiempo de espera en milisegundos para sondear datos de Kafka en ejecutores de trabajos de Spark. El valor predeterminado es `512`.
- `"numRetries"`: (opcional) el número de veces que se reintenta antes de no obtener las compensaciones de Kafka. El valor predeterminado es `3`.
- `"retryIntervalMs"`: (opcional) tiempo en milisegundos para esperar antes de volver a intentar obtener compensaciones Kafka. El valor predeterminado es `10`.
- `"maxOffsetsPerTrigger"`: (opcional) el límite de velocidad en el número máximo de compensaciones que se procesan por intervalo de desencadenador. El número total de compensaciones especificado se divide de forma proporcional entre `topicPartitions` de

diferentes volúmenes. El valor predeterminado es nulo, lo que significa que el consumidor lee todos las compensaciones hasta la última compensación conocida.

- `"minPartitions"`: (opcional) el número mínimo deseado de particiones para leer desde Kafka. El valor predeterminado es nulo, lo que significa que el número de particiones de Spark es igual al número de particiones de Kafka.
- `"includeHeaders"`: (opcional) si se deben incluir los encabezados de Kafka. Cuando la opción se establece en “verdadero”, la salida de datos contendrá una columna adicional denominada “glue_streaming_kafka_headers” con el tipo `Array[Struct(key: String, value: String)]`. El valor predeterminado es “falso”. Esta opción se encuentra disponible en la versión 3.0 o posterior de AWS Glue.
- `"schema"`: (Obligatorio cuando inferSchema se establece en false) Esquema que se va a utilizar para procesar la carga. Si la clasificación es `avro`, el esquema proporcionado debe estar en el formato de esquema Avro. Si la clasificación no es `avro`, el esquema proporcionado debe estar en el formato de esquema DDL.

  A continuación, se muestran algunos ejemplos de esquemas.

  Example in DDL schema format

  ```
  'column1' INT, 'column2' STRING , 'column3' FLOAT
  ```

  Example in Avro schema format

  ```
  {
  "type":"array",
  "items":
  {
  "type":"record",
  "name":"test",
  "fields":
  [
    {
      "name":"_id",
      "type":"string"
    },
    {
      "name":"index",
      "type":
      [
        "int",
  ```

```
      "string",
      "float"
    ]
  }
]
}
}
```

- `"inferSchema"`: (opcional) El valor predeterminado es “false”. Si se establece en “true”, el esquema se detectará durante el tiempo de ejecución desde la carga dentro de `foreachbatch`.
- `"avroSchema"`: (obsoleto) Parámetro utilizado para especificar un esquema de datos Avro cuando se utiliza el formato Avro. Este parámetro se ha quedado obsoleto. Utilice el parámetro `schema`.
- `"addRecordTimestamp"`: (opcional) cuando esta opción se establece en “true”, la salida de datos contendrá una columna adicional denominada “__src_timestamp” que indica la hora en la que el tema recibió el registro correspondiente. El valor predeterminado es "false". Esta opción es compatible con la versión 4.0 o posterior de AWS Glue.
- `"emitConsumerLagMetrics"`: (Opcional) Si la opción se establece en «true», emitirá las métricas correspondientes a cada lote desde el registro más antiguo recibido por el tema hasta el momento en que llegue. AWS Glue CloudWatch El nombre de la métrica es «glue.driver.streaming». maxConsumerLagInMs». El valor predeterminado es "false". Esta opción es compatible con la versión 4.0 o posterior de AWS Glue.

Al escribir, utilice las opciones de conexión con `"connectionType": "kafka"` a continuación:

- `"connectionName"`(Obligatorio) Nombre de la conexión AWS Glue utilizada para conectarse al clúster de Kafka (similar a la fuente de Kafka).
- `"topic"` (Obligatorio) Si existe una columna de temas, su valor se utiliza como el tema al momento de la escritura de la fila en Kafka, a menos que esté establecida la opción de configuración de temas. Es decir, la opción de configuración de `topic` anula la columna de temas.
- `"partition"` (Opcional) Si se especifica un número válido de partición, esta `partition` se utilizará cuando se envíe el registro.

  Si no se especifica ninguna partición pero hay una `key`, se elegirá una partición con el hash de la clave.

  Si no hay ni una `key` ni una `partition`, se elegirá una partición según la partición sticky de esos cambios cuando al menos se produzcan bytes de batch.size en la partición.

- `"key"` (Opcional) Utilizado para la partición si la `partition` es nula.
- `"classification"` (Opcional) El formato de archivo utilizado por los datos en el registro. Solo se admiten los formatos JSON, CSV y Avro.

  Con el formato Avro, podemos brindar un AvroSchema personalizado para la serialización, pero tenga en cuenta que también se tiene que brindar en el origen para la deserialización. De lo contrario, utiliza Apache de forma predeterminada AvroSchema para la serialización.

Además, puede ajustar los receptores de Kafka según sea necesario al actualizar los parámetros de configuración de productor de Kafka. Tenga en cuenta que no hay una lista de permisos en las opciones de conexión; todos los pares valores clave se conservan en el receptor como se encuentran.

Sin embargo, existe una pequeña lista de opciones de rechazos que no tendrá efecto. Para obtener más información, consulte las Configuraciones específicas de Kafka.

Conexión de Kinesis

Puede leer y escribir en Amazon Kinesis Data Streams mediante la información almacenada en una tabla de Catálogo de datos o al brindar información para acceder directamente al flujo de datos. Puede leer la información de Kinesis en un Spark y DataFrame, después, convertirla en un Glue AWS . DynamicFrame Puede DynamicFrames escribir en Kinesis en formato JSON. Si accede directamente a la secuencia de datos, utilice estas opciones para proporcionar información sobre cómo acceder a la secuencia de datos.

Si utiliza `getCatalogSource` o `create_data_frame_from_catalog` para consumir registros de una fuente de streaming de Kinesis, el trabajo tiene la base de datos de Data Catalog y la información del nombre de la tabla, y puede utilizarla para obtener algunos parámetros básicos para la lectura de la fuente de streaming de Kinesis. Si utiliza `getSource`, `getSourceWithFormat`, `createDataFrameFromOptions` o `create_data_frame_from_options` debe especificar estos parámetros básicos mediante las opciones de conexión descritas aquí.

Puede especificar las opciones de conexión para Kinesis al utilizar los siguientes argumentos para los métodos especificados en la clase `GlueContext`.

- Scala
  - `connectionOptions`: se debe utilizar con `getSource`, `createDataFrameFromOptions` y `getSink`

  - `additionalOptions`: se debe utilizar con `getCatalogSource`, `getCatalogSink`
  - `options`: se debe utilizar con `getSourceWithFormat`, `getSinkWithFormat`
- Python
  - `connection_options`: se debe utilizar con `create_data_frame_from_options`, `write_dynamic_frame_from_options`
  - `additional_options`: se debe utilizar con `create_data_frame_from_catalog`, `write_dynamic_frame_from_catalog`
  - `options`: se debe utilizar con `getSource`, `getSink`

Para obtener notas y restricciones sobre los trabajos de ETL de Streaming, consulte the section called “Notas y restricciones de ETL de streaming”.

Configurar Kinesis

Para conectarse a una transmisión de datos de Kinesis en un trabajo de AWS Glue Spark, necesitará algunos requisitos previos:

- Si está leyendo, el trabajo de AWS Glue debe tener permisos de IAM de nivel de acceso de lectura para la transmisión de datos de Kinesis.
- Si está escribiendo, el trabajo de AWS Glue debe tener permisos de IAM de nivel de acceso de escritura para la transmisión de datos de Kinesis.

En algunos casos, tendrá que configurar requisitos previos adicionales:

- Si su trabajo de AWS Glue está configurado con conexiones de red adicionales (normalmente para conectarse a otros conjuntos de datos) y una de esas conexiones proporciona opciones de red de Amazon VPC, esto indicará que su trabajo se comunique a través de Amazon VPC. En este caso, también tendrá que configurar el flujo de datos de Kinesis para que se comunique a través de la VPC de Amazon. Puede hacerlo mediante la creación de un punto de conexión de VPC de tipo interfaz entre la VPC de Amazon y el flujo de datos de Kinesis. Para obtener más información, consulte Uso de Kinesis de Amazon Kinesis Data Streams con puntos de conexión de VPC de interfaz.
- Al especificar Amazon Kinesis Data Streams en otra cuenta, debe configurar los roles y las políticas para permitir el acceso entre cuentas. Para obtener más información, consulte Ejemplo: leer desde un flujo de Kinesis en una cuenta diferente.

Para obtener más información sobre los requisitos previos del trabajo de ETL de Streaming, consulte the section called “Trabajos ETL de streaming”.

Lea desde Kinesis

Ejemplo: lectura de transmisiones desde Kinesis

Se utiliza junto con the section called “forEachBatch”.

Ejemplo de origen de streaming de Amazon Kinesis:

```
kinesis_options =
   { "streamARN": "arn:aws:kinesis:us-east-2:777788889999:stream/fromOptionsStream",
     "startingPosition": "TRIM_HORIZON",
     "inferSchema": "true",
     "classification": "json"
   }
data_frame_datasource0 =
 glueContext.create_data_frame.from_options(connection_type="kinesis",
 connection_options=kinesis_options)
```

Escribir en Kinesis

Ejemplo: escribir en flujos de Kinesis

Se utiliza junto con the section called “forEachBatch”. Se DynamicFrame escribirá en la transmisión en formato JSON. Si el trabajo no se puede escribir después de varios intentos, fallará. De forma predeterminada, cada DynamicFrame registro se enviará a la transmisión de Kinesis de forma individual. Puede configurar este comportamiento mediante `aggregationEnabled` y los parámetros asociados.

Ejemplo de escritura en Amazon Kinesis desde un trabajo de transmisión:

Python

```
glueContext.write_dynamic_frame.from_options(
    frame=frameToWrite
    connection_type="kinesis",
    connection_options={
        "partitionKey": "part1",
        "streamARN": "arn:aws:kinesis:us-east-1:111122223333:stream/streamName",
```

```
    }
)
```

Scala

```
glueContext.getSinkWithFormat(
                connectionType="kinesis",
                options=JsonOptions("""{
                    "streamARN": "arn:aws:kinesis:us-
east-1:111122223333:stream/streamName",
                    "partitionKey": "part1"
                }"""),
            )
            .writeDynamicFrame(frameToWrite)
```

Parámetros de conexión de Kinesis

Designa opciones de conexión para Amazon Kinesis Data Streams.

Utilice las siguientes opciones de conexión para los orígenes de datos de streaming de Kinesis:

- `"streamARN"`: (Obligatorio) Se utiliza para leer/escribir. El ARN de flujo de datos de Kinesis.
- `"classification"`: (Obligatorio para lectura) Se utiliza para leer. El formato de archivo utilizado por los datos del registro. Obligatorio a menos que se proporcione a través del catálogo de datos.
- `"streamName"`: (Opcional) Se usa para leer. Nombre de un flujo de datos de Kinesis para leer. Utilizado con `endpointUrl`.
- `"endpointUrl"`: (Opcional) Se usa para leer. Predeterminado: “https://kinesis.us-east-1.amazonaws.com”. El AWS punto final de la transmisión de Kinesis. No es necesario cambiar esto a menos que se conecte a una región especial.
- `"partitionKey"`: (Opcional) Se usa para escribir. La clave de partición de Kinesis que se utiliza al producir registros.
- `"delimiter"`: (Opcional) Se usa para leer. El separador de valores que se utiliza cuando `classification` es CSV. El valor predeterminado es “`,`”.
- `"startingPosition"`: (Opcional) Se usa para leer. La posición inicial en el flujo de datos de Kinesis para leer los datos. Los valores posibles son `"latest"`, `"trim_horizon"`, `"earliest"` o una cadena de marca de tiempo en formato UTC en el patrón `yyyy-mm-ddTHH:MM:SSZ` (donde `Z` representa un desplazamiento de zona horaria UTC con un +/-. Por ejemplo, “04-04-2023 T

08:00:00-04:00”). El valor predeterminado es `"latest"`. Nota: la cadena Timestamp en formato UTC solo `"startingPosition"` es compatible con la versión 4.0 o posterior de AWS Glue.

- `"failOnDataLoss"`: (Opcional) No se realizará el trabajo si falta o ha caducado alguna partición activa. El valor predeterminado es `"false"`.
- `"awsSTSRoleARN"`: (Opcional) Se usa para escribir/leer. El nombre del recurso de Amazon (ARN) de la función que se va a asumir mediante AWS Security Token Service (AWS STS). Este rol debe tener permisos para describir o leer operaciones de registros del flujo de datos de Kinesis. Debe utilizar este parámetro para acceder a un flujo de datos de otra cuenta. Se utiliza junto con `"awsSTSSessionName"`.
- `"awsSTSSessionName"`: (Opcional) Se usa para escribir/leer. Un identificador para la sesión que asume el rol mediante AWS STS. Debe utilizar este parámetro para acceder a un flujo de datos de otra cuenta. Se utiliza junto con `"awsSTSRoleARN"`.
- `"awsSTSEndpoint"`: (Opcional) El AWS STS punto final que se utilizará al conectarse a Kinesis con un rol asumido. Esto permite usar el AWS STS punto final regional en una VPC, lo que no es posible con el punto final global predeterminado.
- `"maxFetchTimeInMs"`: (Opcional) Se usa para leer. El tiempo máximo que le tomó al ejecutor del trabajo leer los registros del lote actual en el flujo de datos de Kinesis, especificado en milisegundos (ms). Pueden realizarse varias llamadas a la API de `GetRecords` durante este tiempo. El valor predeterminado es `1000`.
- `"maxFetchRecordsPerShard"`: (Opcional) Se usa para leer. El número máximo de registros que se recuperará por partición en el flujo de datos de Kinesis por microlote. Nota: El cliente puede exceder este límite si el trabajo de streaming ya leyó registros adicionales de Kinesis (en la misma llamada de obtención de registros). Si `maxFetchRecordsPerShard` necesita ser preciso, entonces necesita ser un múltiplo de `maxRecordPerRead`. El valor predeterminado es `100000`.
- `"maxRecordPerRead"`: (Opcional) Se usa para leer. El número máximo de registros que se recuperará del flujo de datos de Kinesis en cada operación `getRecords`. El valor predeterminado es `10000`.
- `"addIdleTimeBetweenReads"`: (Opcional) Se usa para leer. Agrega un retardo de tiempo entre dos operaciones `getRecords` consecutivas. El valor predeterminado es `"False"`. Esta opción sólo se puede configurar para Glue versión 2.0 y superior.
- `"idleTimeBetweenReadsInMs"`: (Opcional) Se usa para leer. El tiempo mínimo de retraso entre dos operaciones `getRecords` consecutivas, especificado en ms. El valor predeterminado es `1000`. Esta opción sólo se puede configurar para Glue versión 2.0 y superior.

- `"describeShardInterval"`: (Opcional) Se usa para leer. El intervalo mínimo de tiempo entre dos llamadas a la API `ListShards` para que su script considere cambios en los fragmentos. Para obtener más información, consulte Estrategias para cambios en los fragmentos en la Guía para desarrolladores de Amazon Kinesis Data Streams. El valor predeterminado es `1s`.
- `"numRetries"`: (Opcional) Se usa para leer. El número máximo de reintentos para las solicitudes de la API de Kinesis Data Streams. El valor predeterminado es `3`.
- `"retryIntervalMs"`: (Opcional) Se usa para leer. El periodo de enfriamiento (especificado en ms) antes de volver a intentar la llamada a la API de Kinesis Data Streams. El valor predeterminado es `1000`.
- `"maxRetryIntervalMs"`: (Opcional) Se usa para leer. El periodo de enfriamiento máximo (especificado en ms) entre dos intentos de llamada a la API de Kinesis Data Streams. El valor predeterminado es `10000`.
- `"avoidEmptyBatches"`: (Opcional) Se usa para leer. Evita crear un trabajo de microlotes vacío al comprobar si hay datos no leídos en el flujo de datos de Kinesis antes de que se inicie el lote. El valor predeterminado es `"False"`.
- `"schema"`: (Obligatorio cuando inferSchema se establece en false) Se utiliza para leer. El esquema que se utilizará para procesar la carga útil. Si la clasificación es `avro`, el esquema proporcionado debe estar en el formato de esquema Avro. Si la clasificación no es `avro`, el esquema proporcionado debe estar en el formato de esquema DDL.

  A continuación, se muestran algunos ejemplos de esquemas.

  Example in DDL schema format

  ```
  `column1` INT, `column2` STRING , `column3` FLOAT
  ```

  Example in Avro schema format

  ```
  {
    "type":"array",
    "items":
    {
      "type":"record",
      "name":"test",
      "fields":
      [
        {
          "name":"_id",
  ```

```
        "type":"string"
      },
      {
        "name":"index",
        "type":
        [
          "int",
          "string",
          "float"
        ]
      }
    ]
  }
}
```

- `"inferSchema"`: (Opcional) Se usa para leer. El valor predeterminado es "false". Si se establece en “true”, el esquema se detectará durante el tiempo de ejecución desde la carga dentro de `foreachbatch`.
- `"avroSchema"`: (Obsoleto) Se usa para leer. Parámetro utilizado para especificar un esquema de datos Avro cuando se utiliza el formato Avro. Este parámetro se ha quedado obsoleto. Utilice el parámetro `schema`.
- `"addRecordTimestamp"`: (Opcional) Se usa para leer. Cuando esta opción se establece en “true”, la salida de datos contendrá una columna adicional denominada “__src_timestamp” que indica la hora en la que el flujo recibió el registro correspondiente. El valor predeterminado es "false". Esta opción es compatible con la versión 4.0 o posterior de AWS Glue.
- `"emitConsumerLagMetrics"`: (Opcional) Se usa para leer. Si la opción se establece en «true», para cada lote, emitirá las métricas correspondientes al período comprendido entre el registro más antiguo recibido por la transmisión y el momento en que llegue AWS Glue a CloudWatch él. El nombre de la métrica es «glue.driver.streaming». maxConsumerLagInMs». El valor predeterminado es "false". Esta opción es compatible con la versión 4.0 o posterior de AWS Glue.
- `"fanoutConsumerARN"`: (Opcional) Se usa para leer. El ARN de un consumidor de un flujo de Kinesis para el flujo especificado en `streamARN`. Se utiliza para habilitar el modo de distribución mejorada para la conexión de Kinesis. Para obtener más información sobre cómo consumir una transmisión de Kinesis con una distribución mejorada, consulte the section called “Uso de una distribución mejorada en los trabajos de streaming de Kinesis”.
- `"recordMaxBufferedTime"`: (Opcional) Se usa para escribir. Predeterminado: 1000 (ms). Tiempo máximo que un registro permanece almacenado en búfer mientras espera a ser escrito.

- `"aggregationEnabled"`: (Opcional) Se usa para escribir. Valor predeterminado: verdadero. Especifica si los registros deben agregarse antes de enviarlos a Kinesis.
- `"aggregationMaxSize"`: (Opcional) Se usa para escribir. Predeterminado: 51 200 (bytes). Si un registro supera este límite, omitirá el agregador. Nota: Kinesis impone un límite de 50 KB en el tamaño del registro. Si lo establece por encima de 50 KB, Kinesis rechazará los registros de gran tamaño.
- `"aggregationMaxCount"`: (Opcional) Se usa para escribir. Predeterminado: 4294967295. Número máximo de elementos a empaquetar en un registro agregado.
- `"producerRateLimit"`: (Opcional) Se usa para escribir. Predeterminado: 150 (%). Limita el rendimiento por partición enviado desde un solo productor (por ejemplo, su trabajo), como porcentaje del límite de backend.
- `"collectionMaxCount"`: (Opcional) Se usa para escribir. Predeterminado: 500. Número máximo de artículos para incluir en una PutRecords solicitud.
- `"collectionMaxSize"`: (Opcional) Se usa para escribir. Predeterminado: 5 242 880 (bytes). Cantidad máxima de datos que se pueden enviar con una PutRecords solicitud.

AWS Glue Opciones de streaming

Designa una conexión a un clúster de Kafka o a un clúster de Amazon Managed Streaming for Apache Kafka.

Puede leer y escribir en los flujos de datos de Kafka con la información almacenada en la tabla del Catálogo de datos o al brindar la información para acceder de manera directa al flujo de datos. Puedes leer información de Kafka en una Spark DataFrame y luego convertirla en una AWS Glue DynamicFrame. Puedes escribir DynamicFrames en Kafka en formato JSON. Si accede directamente a la secuencia de datos, utilice estas opciones para proporcionar información sobre cómo acceder a la secuencia de datos.

Si utiliza `getCatalogSource` o `create_data_frame_from_catalog` para consumir los registros de un origen de streaming de Kafka, o `getCatalogSink` o `write_dynamic_frame_from_catalog` para escribir registros en Kafka, el trabajo cuenta con la base de datos del Catálogo de datos y la información del nombre de la tabla, lo cual se puede utilizar para obtener algunos parámetros básicos para la lectura de un origen de streaming de Kafka. Si utiliza `getSource`, `getCatalogSink`, `getSourceWithFormat`, `getSinkWithFormat`, `createDataFrameFromOptions`, `create_data_frame_from_options`

o `write_dynamic_frame_from_catalog`, debe especificar estos parámetros básicos con las opciones de conexión que se describen aquí.

Puede especificar las opciones de conexión para Kafka con los argumentos que se mencionan a continuación para los métodos especificados en la clase `GlueContext`.

- Scala
  - `connectionOptions`: se debe utilizar con `getSource`, `createDataFrameFromOptions` y `getSink`
  - `additionalOptions`: se debe utilizar con `getCatalogSource`, `getCatalogSink`
  - `options`: se debe utilizar con `getSourceWithFormat`, `getSinkWithFormat`
- Python
  - `connection_options`: se debe utilizar con `create_data_frame_from_options`, `write_dynamic_frame_from_options`
  - `additional_options`: se debe utilizar con `create_data_frame_from_catalog`, `write_dynamic_frame_from_catalog`
  - `options`: se debe utilizar con `getSource`, `getSink`

Para obtener notas y conocer las restricciones sobre los trabajos de ETL de transmisión, consulte the section called “Notas y restricciones de ETL de streaming”.

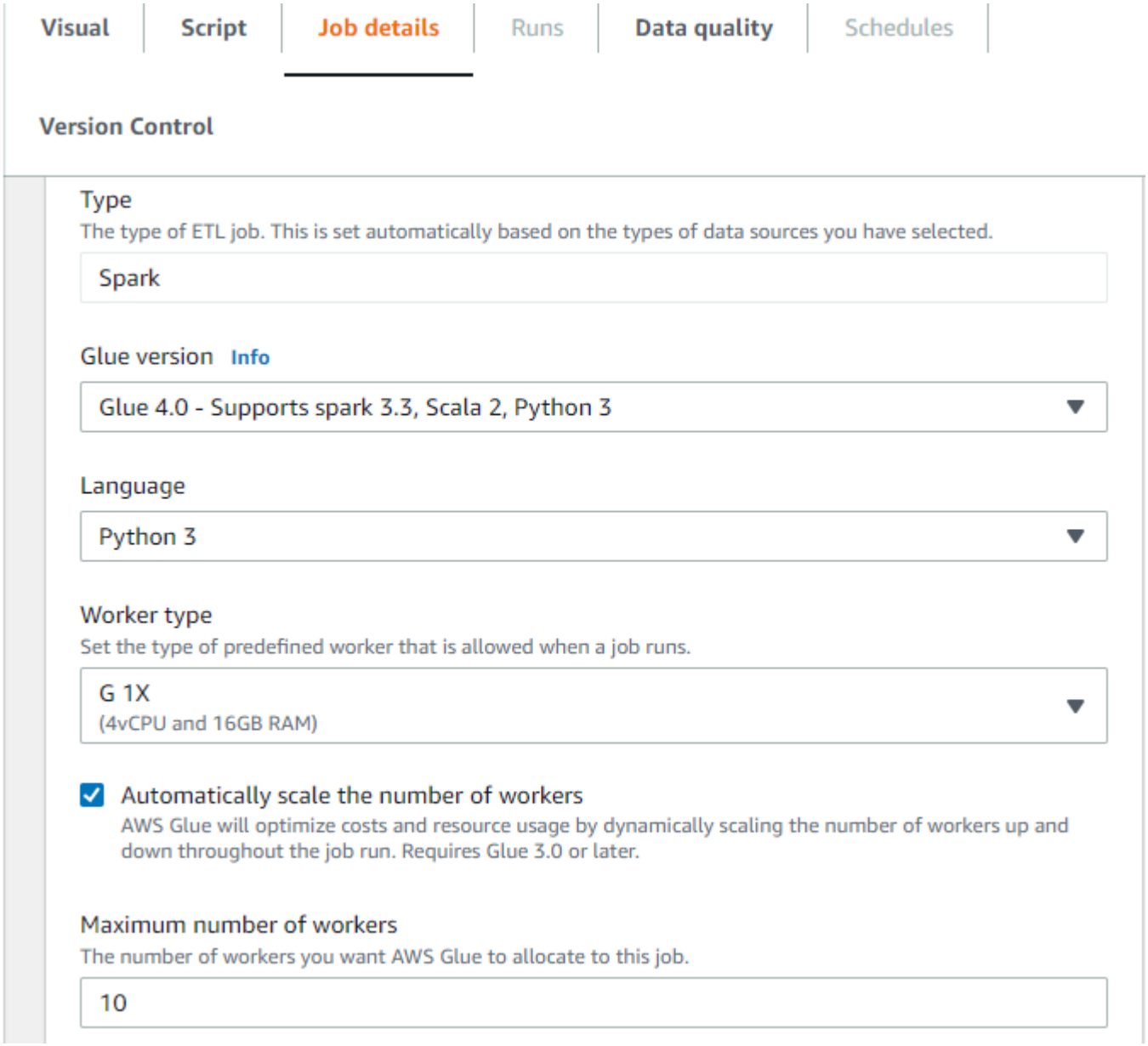
Escalado automático de la transmisión de AWS Glue

En las siguientes secciones, se ofrece información acerca del escalado automático de la transmisión de AWS Glue.

Habilitar Auto Scaling en AWS Glue Studio

En la página Job details (Detalles del trabajo) en AWS Glue Studio, elija el tipo como Spark o Spark Streaming y Glue version (Versión de Glue) como **`Glue 3.0`** o **`Glue 4.0`**. A continuación, se mostrará una casilla de verificación debajo de Worker type (Tipo de empleado).

- Seleccione la opción Automatically scale the number of workers (Escalar automáticamente el número de empleados).
- Establezca el Maximum number of workers (Número máximo de empleados) para definir el número máximo de empleados que se pueden asignar a la ejecución de trabajos.



Habilitación de Auto Scaling con AWS CLI o SDK

Para habilitar Auto Scaling desde la AWS CLI para la ejecución de un trabajo, ejecute `start-job-run` con la siguiente configuración:

```
{
    "JobName": "<your job name>",
    "Arguments": {
        "--enable-auto-scaling": "true"
    },
    "WorkerType": "G.2X", // G.1X and G.2X are allowed for Auto Scaling Jobs
```

```
    "NumberOfWorkers": 20, // represents Maximum number of workers
    ...other job run configurations...
}
```

Una vez que haya finalizado la ejecución de un trabajo de ETL, también puede llamar a `get-job-run` para comprobar el uso real de recursos de la ejecución del trabajo en segundos de DPU. Nota: El nuevo campo DPUSeconds solo se mostrará para los trabajos por lotes en AWS Glue 3.0 o posterior con el escalado automático habilitado. Este campo no es compatible con trabajos de streaming.

```
$ aws glue get-job-run --job-name your-job-name --run-id jr_xx --endpoint https://
glue.us-east-1.amazonaws.com --region us-east-1
{
    "JobRun": {
        ...
        "GlueVersion": "3.0",
        "DPUSeconds": 386.0
    }
}
```

También se pueden configurar ejecuciones de trabajos con Auto Scaling mediante el AWS Glue SDK con la misma configuración.

Cómo funcionan

Escalado entre microlotes

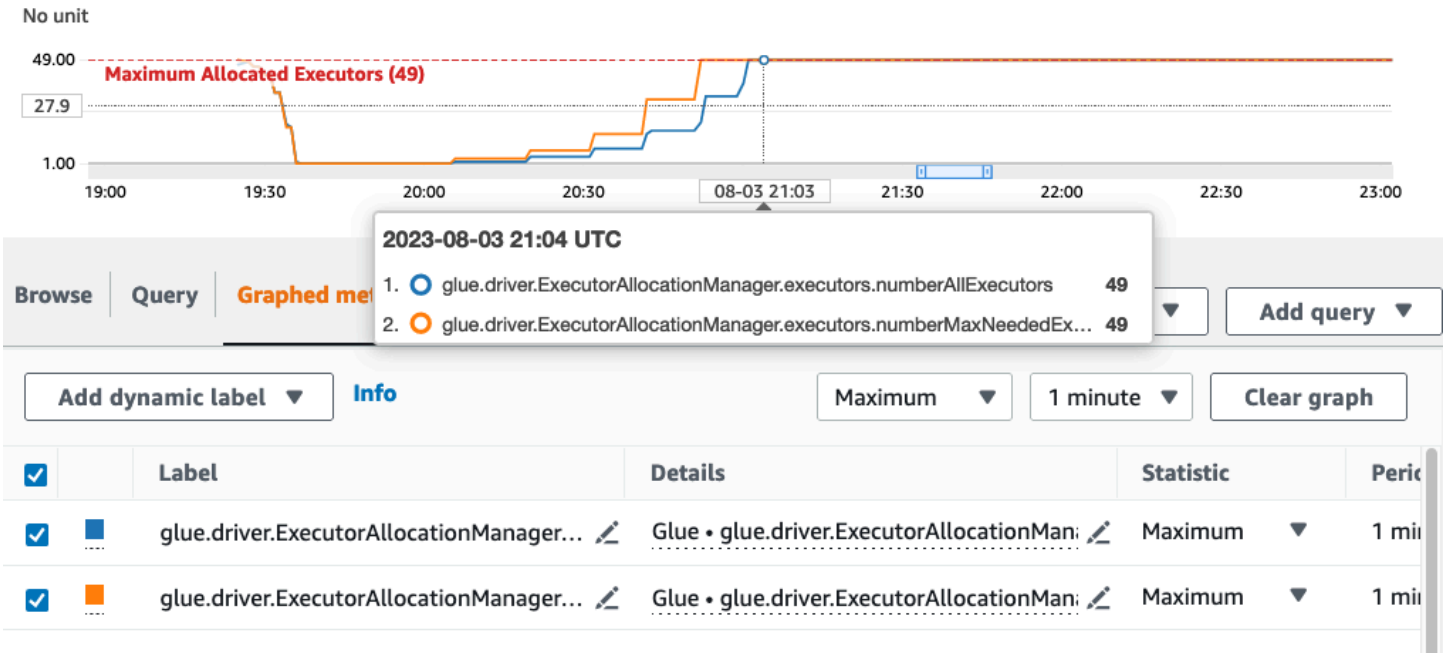
El siguiente ejemplo se usa para describir cómo funciona el escalado automático.

- Tiene un trabajo de AWS Glue que comienza con 50 DPU.
- El escalado automático está habilitado.

En este ejemplo, AWS Glue analiza la métrica “batchProcessingTimeInMs” de algunos microlotes y determina si los trabajos se están completando dentro del tamaño de ventana que ha establecido. Si sus trabajos se completan antes y en función de la rapidez con la que se terminen, es posible que AWS Glue se reduzca verticalmente. Esta métrica, trazada con “numberAllExecutors”, se puede supervisar en Amazon CloudWatch para ver cómo funciona el escalado automático.

El número de ejecutores se escala o reduce verticalmente de manera exponencial solo después de que se complete cada microlote. Como puede ver en el registro de supervisión de Amazon

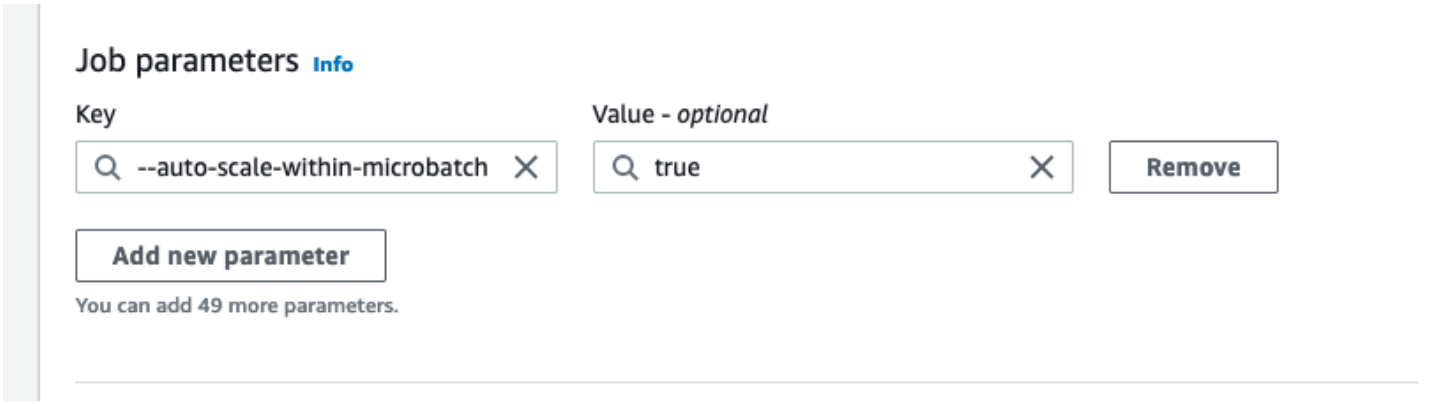
CloudWatch, AWS Glue analiza la cantidad de ejecutores necesarios (línea naranja) y escala los ejecutores (línea azul) para que coincidan automáticamente con esa cantidad.



Una vez que AWS Glue reduzca verticalmente el número de ejecutores y observe que los volúmenes de datos aumentan y, en consecuencia, aumenta el tiempo de procesamiento de los microlotes, AWS Glue escalará verticalmente hasta los 50 DPU, que es el límite máximo especificado.

Escalado dentro de microlotes

En el ejemplo anterior, el sistema supervisa algunos microlotes completados para tomar una decisión acerca de si escalarlos o reducirlos verticalmente. Las ventanas más largas requieren un escalado automático para responder de manera más rápida dentro del microlote en vez de esperar unos pocos microlotes. En estos casos, puede usar la configuración adicional `--auto-scaIe-within-microbatch` como `true`. Puede agregarla a las propiedades del trabajo de AWS Glue en AWS Glue Studio como se muestra a continuación.



Ventanas de mantenimiento para AWS Glue Streaming

AWS Glue realiza periódicamente actividades de mantenimiento. Durante estos períodos de mantenimiento, AWS Glue tendrás que reiniciar tus trabajos de streaming. Puede controlar cuándo se reinician los trabajos especificando los períodos de mantenimiento. En esta sección, explicamos dónde puede configurar el período de mantenimiento y los comportamientos específicos que debe tener en cuenta.

Temas

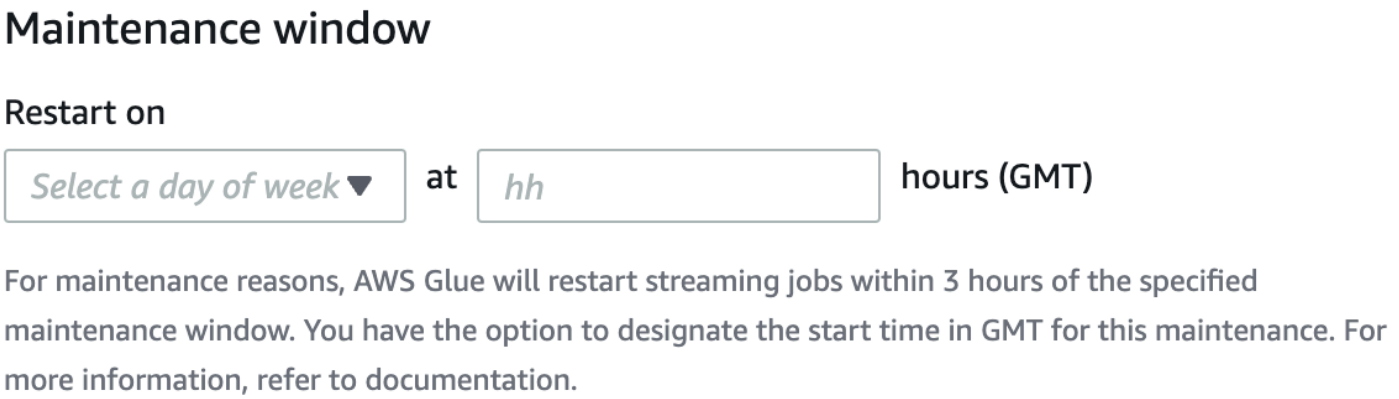
- Configuración de una ventana de mantenimiento
- El comportamiento de la ventana de mantenimiento
- Supervisión de trabajos
- Gestión de la pérdida de datos

Configuración de una ventana de mantenimiento

Puede configurar una ventana de mantenimiento mediante AWS Glue Studio o las API.

Configurar una ventana de mantenimiento en AWS Glue Studio

Puede especificar una ventana de mantenimiento en la página de detalles del trabajo de su trabajo de AWS Glue streaming. Puede especificar el día y la hora en GMT. AWS Glue reiniciará el trabajo dentro del intervalo de tiempo especificado.



Configurar una ventana de mantenimiento en la API

También puedes configurar la ventana de mantenimiento en la API Create Job. A continuación, se muestra un ejemplo de configuración de una ventana de mantenimiento mediante la API.

```
aws glue create-job –name jobName –role roleArnForTheJob –command
 Name=gluestreaming,ScriptLocation=s3-path-to-the-script --maintenance-window="Sun:10"
```

Un ejemplo de comando es el siguiente:

```
aws glue create-job –name testMaintenance –role arn:aws:iam::012345678901:role/
Glue_DefaultRole –command Name=gluestreaming,ScriptLocation=s3://glue-example-test/
example.py –maintenance-window="Sun:10
```

El comportamiento de la ventana de mantenimiento

AWS Glue sigue una serie de pasos para decidir cuándo reiniciar un trabajo:

1. Cuando se inicia un nuevo trabajo de streaming, AWS Glue primero comprueba si hay un tiempo de espera asociado a la ejecución del trabajo. Un tiempo de espera permite configurar la hora de finalización del trabajo. Si el tiempo de espera es inferior a 7 días, el trabajo no se reiniciará.
2. Si el tiempo de espera es superior a 7 días, AWS Glue comprueba si el período de mantenimiento está configurado para el trabajo. Si es así, se abre esa ventana y se asigna a la ejecución del trabajo. AWS Glue reiniciará el trabajo en un plazo de 3 horas a partir del período de mantenimiento especificado. Por ejemplo, si configura el período de mantenimiento para el lunes a las 10:00 a.m. GMT, sus trabajos se reiniciarán entre las 10:00 a.m. GMT y las 13:00 p.m. GMT.
3. Si el período de mantenimiento no está configurado, establece AWS Glue automáticamente la hora de reinicio en 7 días después de la hora de inicio de la ejecución de la tarea. Por ejemplo, si inició el trabajo el 1 de julio de 2024 a las 12:00 a. m. GMT y no especificó los períodos de mantenimiento, el trabajo se configurará para que se reinicie el 8 de julio de 2024 a las 12:00 a. m. GMT.

> **Note**
>
> Si ya estás realizando tareas de streaming, este cambio te afectará a partir del 1 de julio de 2024. Tendrá tiempo hasta el 30 de junio para configurar sus períodos de mantenimiento. Después del 1 de julio, todos los trabajos de streaming que inicies se

reiniciarán según esta documentación. Si necesita asistencia adicional, puede ponerse en contacto con AWS Support.

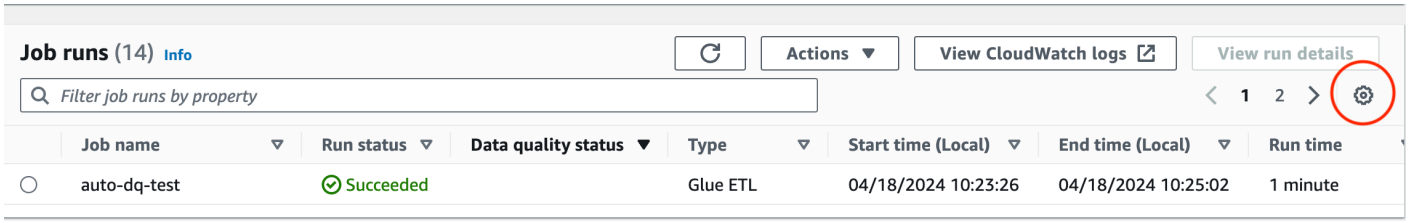
4. A veces, es AWS Glue posible que no pueda reiniciar el trabajo, especialmente cuando el microlote en curso no se procesa. En estos casos, el trabajo no se interrumpirá. En estos casos, AWS Glue reiniciará el trabajo después de 14 días y, en este caso, no se respetará el período de mantenimiento.

Supervisión de trabajos

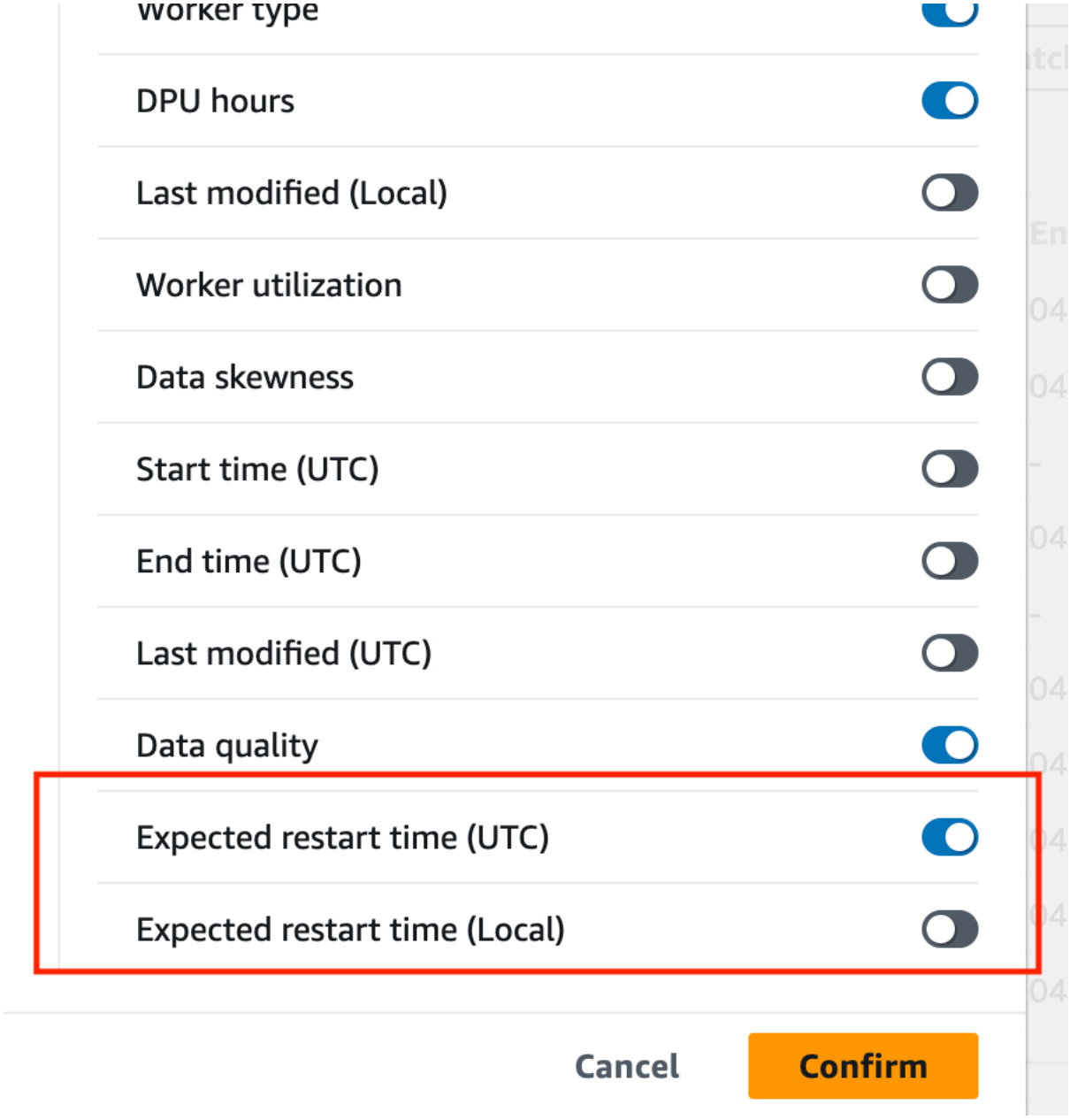
Puede supervisar los trabajos en la página AWS Glue Studio Monitoring.

Para ver la hora prevista para el próximo reinicio de los trabajos de streaming, muestre la columna de la tabla Ejecuciones de trabajos en la página Supervisión.

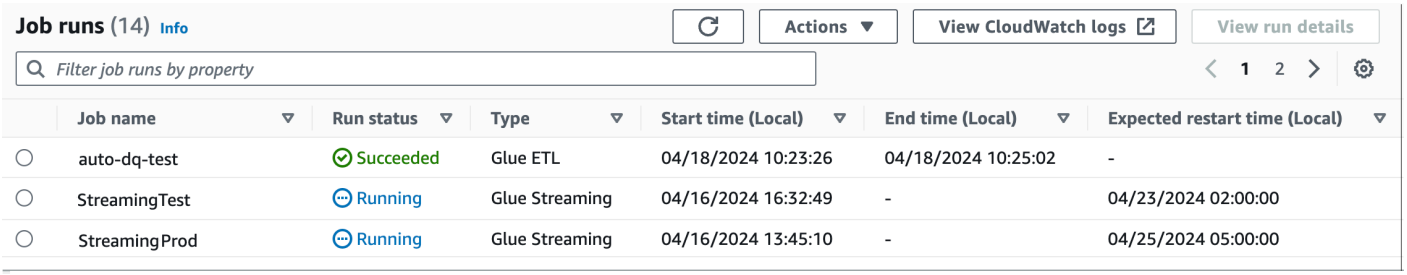
1. Haga clic en el icono con forma de engranaje situado en la parte superior derecha de la tabla.



2. Desplázate hacia abajo y activa la columna Tiempo de reinicio esperado. Están disponibles las opciones UTC y hora local.



3. A continuación, puede ver las columnas de la tabla.



El trabajo original tendrá el estado «CADUCADO» y la nueva instancia de trabajo tendrá el estado «EN EJECUCIÓN». La nueva ejecución del trabajo que se reinició tendrá un ID de ejecución de trabajo como una concatenación del ID de ejecución del trabajo inicial más el prefijo «restart_», que representa el recuento de reinicios. Por ejemplo, si el ID de ejecución del trabajo inicial es`jr_1234`, la ejecución del trabajo reiniciada tendrá el ID del primer reinicio. `jr1234_restart_1` El segundo reinicio será `jr1234_restart_2` para el segundo reinicio y así sucesivamente.

El reintento no se verá afectado por los reinicios. Si se produce un error en una ejecución y se inicia una nueva debido a un reintento automático, el contador de reinicios volverá a empezar desde 1. Por ejemplo, si se produce un error en una ejecución`jr_1234_attempt_3_restart_5`, al reintentarlo automáticamente se iniciará una nueva ejecución con un identificador; `jr_id1_attempt_4` y si este intento se reinicia transcurridos 7 días, se iniciará con el nuevo identificador de ejecución. `jr_id1_attempt_4_restart_1`

Gestión de la pérdida de datos

Durante los reinicios por mantenimiento, AWS Glue Streaming sigue un proceso que garantiza la integridad y la coherencia de los datos entre la ejecución del trabajo anterior y la ejecución del trabajo reiniciado. Tenga en cuenta que AWS Glue esto no garantiza la integridad ni la coherencia de los datos entre los reinicios de los trabajos, por lo que recomendamos tener en cuenta la arquitectura para gestionar los datos duplicados en los trabajos de streaming.

1. Detección de las condiciones de reinicio por mantenimiento: la AWS Glue transmisión monitorea las condiciones que indican cuándo debe activarse un reinicio por mantenimiento, por ejemplo, cuando se alcanza un período de mantenimiento después de 7 días o si es necesario reiniciar por completo después de 14 días.
2. Invocar una finalización correcta: cuando se cumplen las condiciones de reinicio por mantenimiento, AWS Glue Streaming inicia un proceso de finalización correcta para la tarea que se está ejecutando actualmente. Este proceso incluye los siguientes pasos:
   a. Detener la ingesta de nuevos datos: el trabajo de streaming deja de consumir nuevos datos de las fuentes de entrada (por ejemplo, temas de Kafka, transmisiones de Kinesis o archivos).
   b. Procesamiento de datos pendientes: el trabajo continúa procesando todos los datos que ya estén presentes en sus búferes o colas internos.
   c. Consignación de compensaciones y puntos de control: el trabajo envía las últimas compensaciones o puntos de control a sistemas externos (por ejemplo, Kafka, Kinesis o Amazon S3) para garantizar que la tarea reiniciada pueda continuar desde donde la dejó la anterior.

3. Reiniciar el trabajo: una vez finalizado el proceso de finalización correcto, Streaming reinicia el trabajo utilizando el estado y los puntos de control conservados AWS Glue . El trabajo reiniciado retoma el procesamiento desde el último punto de control o compensación comprometido, lo que garantiza que no se pierdan ni dupliquen datos.
4. Reanudación del procesamiento de datos: el trabajo reiniciado reanuda el procesamiento de datos desde el punto en el que lo dejó el trabajo anterior. Continúa incorporando nuevos datos de las fuentes de entrada, empezando por el último desfase o punto de control registrado, y procesa los datos de acuerdo con la lógica ETL definida.

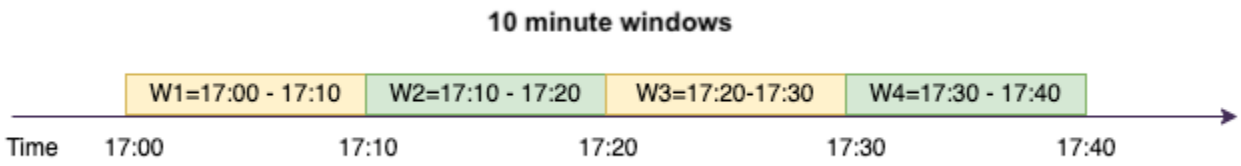
Conceptos avanzados de la transmisión de AWS Glue

En las aplicaciones contemporáneas basadas en datos, la importancia de los datos disminuye con el tiempo y su valor pasa de ser predictivo a reactivo. Como resultado, los clientes desean procesar los datos en tiempo real para tomar decisiones más rápidas. Cuando se trata de fuentes de datos en tiempo real, como las de los sensores de IoT, es posible que los datos lleguen desordenados o que se produzcan retrasos en el procesamiento debido a la latencia de la red y a otros fallos relacionados con el origen durante la ingesta. Como parte de la plataforma de AWS Glue, la transmisión de AWS Glue aprovecha estas capacidades para ofrecer ETL de transmisión escalable y sin servidor, con tecnología de transmisión estructurado de Apache Spark, que permite a los usuarios procesar datos en tiempo real.

En este tema, exploraremos los conceptos y capacidades avanzados de la transmisión de AWS Glue.

Consideraciones de tiempo a la hora de procesar los trabajos de transmisión

Al procesar los trabajos de transmisión, hay cuatro nociones de tiempo:



- Hora del evento: la marca temporal de cuando se produjo el evento. En la mayoría de los casos, este campo está incrustado en los propios datos del evento, en el origen.

- E vent-time-window: El intervalo de tiempo entre dos tiempos de eventos. Como se muestra en el diagrama anterior, W1 es event-time-window entre las 17:00 y las 17:10. Cada uno event-time-window es una agrupación de varios eventos.
- Hora de activación: la hora de activación controla la frecuencia con la que se procesan los datos y se actualizan los resultados. Este es el momento en que el procesamiento del microlote comenzó.
- Tiempo de ingesta: el momento en que los datos de transmisión se ingirieron en el servicio de transmisión. Si la hora del evento no está integrada en el propio evento, en algunos casos se puede usar como ventana.

Creación de ventanas

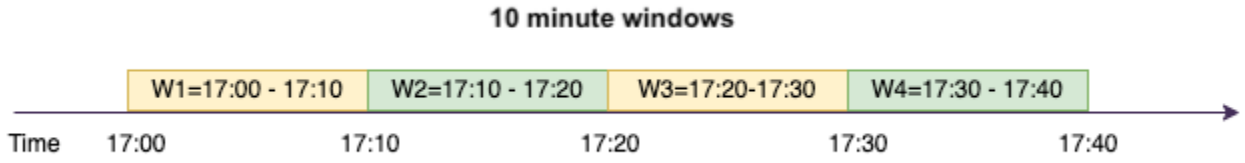
La creación de ventanas es una técnica en la que se agrupan y agregan varios eventos por. event-time-window En los siguientes ejemplos, analizaremos las ventajas de la creación de ventanas y cuándo usarla.

Según el caso de uso empresarial, Spark admite tres tipos de ventanas de tiempo.

- Ventana giratoria: una serie de tamaños fijos que no se superponen event-time-windows sobre la que se agregan.
- Ventana deslizante: son similares a las ventanas de saltos de tamaño constante, ya que tienen un “tamaño fijo”, pero las ventanas pueden superponerse o deslizarse siempre que la duración del deslizamiento sea inferior a la duración de la propia ventana.
- Ventana de sesión: comienza con un evento de datos de entrada y continúa expandiéndose mientras reciba información dentro de un intervalo o periodo de inactividad. Una ventana de sesión puede tener un tamaño estático o dinámico igual a la longitud de la ventana, en función de las entradas.

Ventana de saltos de tamaño constante

La ventana giratoria es una serie de tamaños event-time-windows fijos que no se superponen y que se agregan. Vamos a entender esto con un ejemplo del mundo real.



La empresa ABC Auto desea realizar una campaña de marketing para una nueva marca de coches deportivos. Quiere elegir una ciudad donde haya la mayor cantidad de fanáticos de los coches deportivos. Para lograr este objetivo, muestra un breve anuncio de 15 segundos que presenta el coche en su sitio web. Todos los «clics» y la «ciudad» correspondiente se graban y se transmiten. Amazon Kinesis Data Streams Queremos contar el número de clics en un intervalo de 10 minutos y agruparlo por ciudad para ver qué ciudad tiene la mayor demanda. El siguiente valor es el resultado de la agregación.

window_start_time	window_end_time	ciudad	total_clicks
07-07-2020 17:00:00	2023-07-10 17:10:00	Dallas	75
2023-07-10 17:00:00	2023-07-10 17:10:00	Chicago	10
2023-07-10 17:20:00	2023-07-10 17:30:00	Dallas	20
2023-07-10 17:20:00	2023-07-10 17:30:00	Chicago	50

Como se ha explicado anteriormente, event-time-windows son diferentes de los intervalos de tiempo de activación. Por ejemplo, aunque el tiempo de activación sea cada minuto, los resultados de la salida solo mostrarán ventanas de agregación de 10 minutos que no se superponen. Para la optimización, es mejor alinear el intervalo de activación con el event-time-window.

En la tabla anterior, Dallas registró 75 clics en la ventana de 17:00 h a 17:10 h, mientras que Chicago tuvo 10 clics. Además, no hay datos para la ventana de 17:10 h a 17:20 h de ninguna ciudad, por lo que se omite esta ventana.

Ahora puede realizar un análisis más detallado de estos datos en la aplicación de análisis posterior para determinar cuál es la mejor ciudad para llevar a cabo la campaña de marketing.

Uso de ventanas de saltos de tamaño constante en AWS Glue

1. Crea un Amazon Kinesis Data Streams DataFrame y lee desde él. Ejemplo:

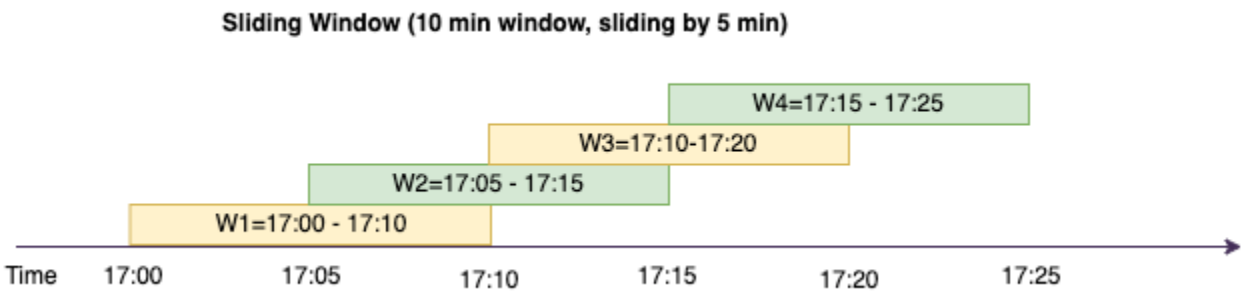
```
parsed_df = kinesis_raw_df \
                .selectExpr('CAST(data AS STRING)') \
                .select(from_json("data", ticker_schema).alias("data")) \
                .select('data.event_time','data.ticker','data.trade','data.volume',
 'data.price')
```

2. Procese los datos en una ventana de saltos de tamaño constante. En el siguiente ejemplo, los datos se agrupan en función del campo de entrada “event_time” en ventanas de tamaño constante de 10 minutos y escriben la salida en un lago de datos de Amazon S3.

```
grouped_df = parsed_df \
                .groupBy(window("event_time", "10 minutes"), "city") \
                .agg(sum("clicks").alias("total_clicks"))

summary_df = grouped_df \
                .withColumn("window_start_time", col("window.start")) \
                .withColumn("window_end_time", col("window.end")) \
                .withColumn("year", year("window_start_time")) \
                .withColumn("month", month("window_start_time")) \
                .withColumn("day", dayofmonth("window_start_time")) \
                .withColumn("hour", hour("window_start_time")) \
                .withColumn("minute", minute("window_start_time")) \
                .drop("window")

write_result = summary_df \
                .writeStream \
                .format("parquet") \
                .trigger(processingTime="10 seconds") \
                .option("checkpointLocation", "s3a://bucket-stock-stream/stock-
stream-catalog-job/checkpoint/") \
                .option("path", "s3a://bucket-stock-stream/stock-stream-catalog-
job/summary_output/") \
                .partitionBy("year", "month", "day") \
                .start()
```

Ventana deslizante

Las ventanas deslizantes son similares a las de saltos de tamaño constante, ya que tienen un “tamaño fijo”, pero las ventanas pueden superponerse o deslizarse siempre que la duración del deslizamiento sea menor que la duración de la propia ventana. Debido a la naturaleza del deslizamiento, una entrada se puede vincular a varias ventanas.



Para entenderlo mejor, consideremos el ejemplo de un banco que quiere detectar posibles fraudes con tarjetas de crédito. Una aplicación de transmisión podría supervisar un flujo continuo de transacciones con tarjetas de crédito. Estas transacciones podrían agruparse en ventanas de 10 minutos de duración y, cada 5 minutos, la ventana se deslizaría hacia adelante, eliminando los 5 minutos de datos más antiguos y agregando los últimos 5 minutos de datos nuevos. Dentro de cada ventana, las transacciones podrían agruparse por país para detectar patrones sospechosos, como una transacción en EE. UU. seguida inmediatamente de otra en Australia. Para simplificar, clasificaremos estas transacciones como fraude cuando el importe total de las transacciones sea superior a 100 USD. Si se detecta un patrón de este tipo, es señal de un posible fraude y la tarjeta puede congelarse.

El sistema de procesamiento de tarjetas de crédito envía una serie de transacciones a Kinesis para cada identificador de tarjeta junto con el país. Un AWS Glue trabajo ejecuta el análisis y produce la siguiente salida agregada.

window_start_time	window_end_time	card_last_four	país	total_amount
2023-07-10 17:00:00	2023-07-10 17:10:00	6544	EE. UU.	85
2023-07-10 17:00:00	2023-07-10 17:10:00	6544	Australia	10
2023-07-10 17:05:45	2023-07-10 17:15:45	6544	EE. UU.	50
2023-07-10 17:10:45	2023-07-10 17:20:45	6544	EE. UU.	50

window_start_time	window_end_time	card_last_four	país	total_amount
2023-07-10 17:10:45	2023-07-10 17:20:45	6544	Australia	150

Según la agregación anterior, puede ver que el intervalo de 10 minutos se desliza cada 5 minutos, sumado por el importe de la transacción. La anomalía se detecta en el intervalo de 17:10 h a 17:20 h, donde hay un valor atípico, es decir, una transacción de 150 USD en Australia. AWS Glue puede detectar esta anomalía y activar un evento de alarma con la clave infractora en un tema de SNS mediante boto3. Además, una función Lambda puede suscribirse a este tema y tomar medidas.

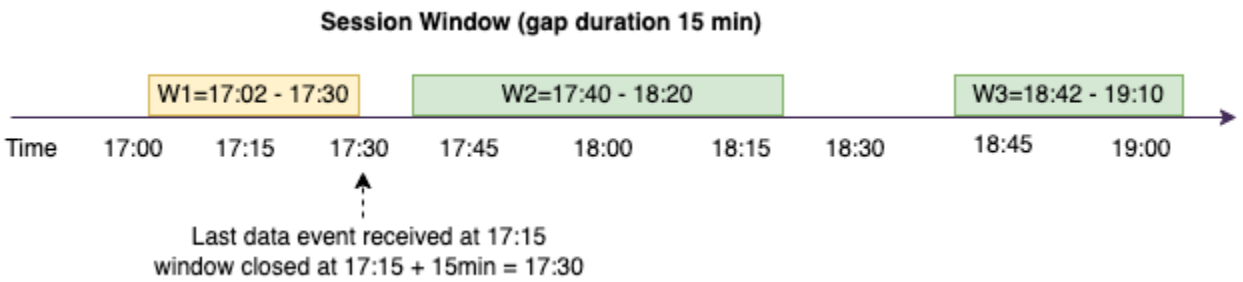
Datos de proceso en una ventana deslizante

La cláusula `group-by` y la función de ventana se usan para implementar la ventana deslizante, como se muestra a continuación.

```
grouped_df = parsed_df \
            .groupBy(window(col("event_time"), "10 minute", "5 min"), "country",
 "card_last_four") \
            .agg(sum("tx_amount").alias("total_amount"))
```

Ventana de sesión

A diferencia de las dos ventanas anteriores, que tienen un tamaño fijo, la ventana de sesión puede tener un tamaño estático o dinámico de la longitud de la ventana, según las entradas. Una ventana de sesión comienza con un evento de datos de entrada y continúa expandiéndose mientras reciba información dentro de un intervalo o periodo de inactividad.



Tomemos un ejemplo. La empresa ABC Hotel quiere saber cuál es el momento más concurrido de la semana y ofrecer mejores ofertas a sus huéspedes. En cuanto un invitado se registra, se abre una ventana de sesión y Spark mantiene un estado de agregación correspondiente. event-time-window Cada vez que un huésped se registra, se genera un evento al que se envía. Amazon Kinesis Data Streams El hotel decide cerrar el establecimiento si no se event-time-window puede hacer el registro de entrada durante un período de 15 minutos. El siguiente event-time-window volverá a empezar cuando haya un nuevo registro. El resultado es el siguiente.

window_start_time	window_end_time	ciudad	total_checkins
2023-07-10 17:02:00	2023-07-10 17:30:00	Dallas	50
2023-07-10 17:02:00	2023-07-10 17:30:00	Chicago	25
2023-07-10 17:40:00	2023-07-10 18:20:00	Dallas	75
2023-07-10 18:50:45	2023-07-10 19:15:45	Dallas	20

El primer registro se produjo con event_time a las 17:02 h. La agregación comenzará a las 17:02. event-time-window Esta agregación continuará mientras recibamos los eventos dentro de los 15 minutos de duración. En el ejemplo anterior, el último evento que recibimos fue a las 17:15 h y, durante los siguientes 15 minutos, no hubo ningún evento. Como resultado, Spark la cerró event-time-window a las 17:15 +15 min = 17:30 y la estableció entre las 17:02 y las 17:30. Comenzó de nuevo a las event-time-window 17:47 cuando recibió un nuevo evento de datos de registro.

Datos de proceso en una ventana de sesión

La cláusula `group-by` y la función de ventana se usan para implementar la ventana deslizante.

```
grouped_df = parsed_df \
            .groupBy(session_window(col("event_time"), "10 minute"), "city") \
            .agg(count("check_in").alias("total_checkins"))
```

Modos de salida

El modo de salida es el modo en el que los resultados de la tabla ilimitada se escriben en el receptor externo. Hay tres modos disponibles. En el siguiente ejemplo, contamos las apariciones de una palabra a medida que se transmiten y se procesan líneas de datos en cada microlote.

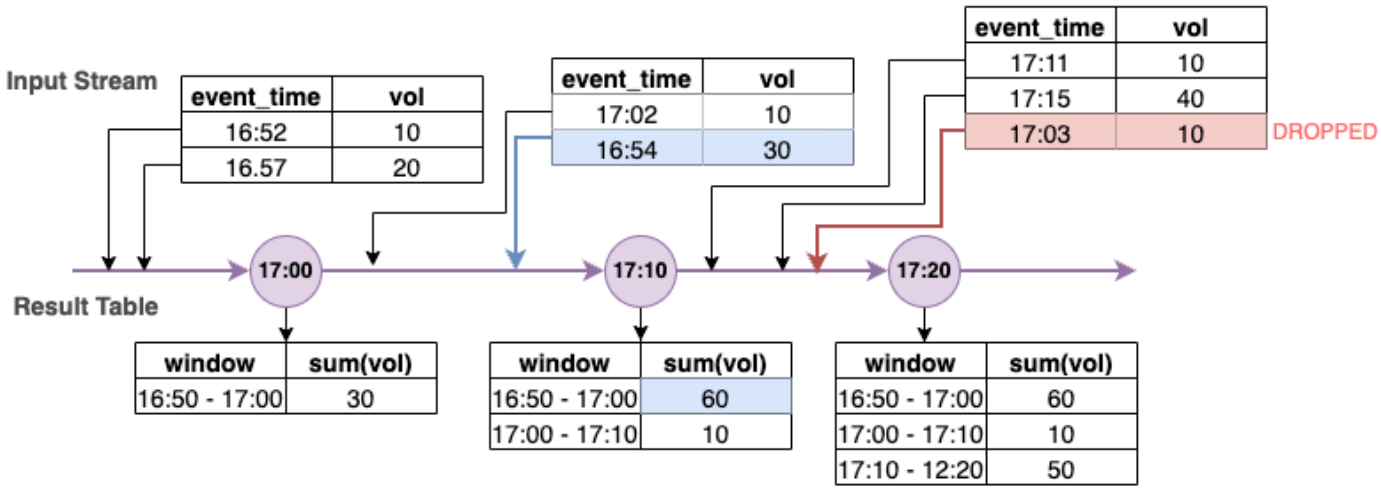
- Modo completo: toda la tabla de resultados se escribirá en el colector después de cada microprocesamiento por lotes, aunque el recuento de palabras no se haya actualizado en el momento actual event-time-window.
- Modo de adición: este es el modo predeterminado, en el que solo se escribirán en el receptor las palabras o filas nuevas que se hayan agregado a la tabla de resultados desde la última activación. Este modo es adecuado para la transmisión sin estado de consultas como map, flatMap, filter, etc.
- Modo de actualización: solo se escribirán en el receptor las palabras o filas de la tabla de resultados que se hayan actualizado o agregado desde la última activación.

> **Note**
>
> Modo de salida = las ventanas de sesión no admiten la opción de actualización.

Manejo de datos tardíos y marcas de agua

Cuando se trabaja con datos en tiempo real, es posible que se produzcan retrasos en la llegada de los datos debido a la latencia de la red y a fallos en las fases iniciales, por lo que necesitamos un mecanismo que permita volver a realizar la agregación cuando no event-time-window lleguen. Sin embargo, para ello, es necesario mantener el estado. Al mismo tiempo, es necesario limpiar los datos más antiguos para limitar el tamaño del estado. En la versión 2.1 de Spark se agregó compatibilidad con una característica llamada “marca de agua” que mantiene el estado y permite al usuario especificar el umbral para los datos tardíos.

Con referencia al ejemplo anterior acerca de la cotización bursátil, consideremos que el límite permitido para los datos tardíos es de no más de 10 minutos. Para simplificar las cosas, supondremos que es una ventana de saltos de tamaño constante, que la cotización es AMZ y que la operación es BUY.



En el diagrama anterior, calculamos el volumen total en una ventana de saltos de tamaño constante de 10 minutos. Tenemos la activación a las 17:00 h, a las 17:10 h y a las 17:20 h. Sobre la flecha de la línea temporal, tenemos el flujo de datos de entrada y debajo está la tabla de resultados ilimitada.

En la primera ventana de saltos de tamaño constante de 10 minutos, agregamos en función de event_time, y total_volume se calculó como 30. En el segundo event-time-window, spark obtuvo el primer evento de datos con event_time= 17:02. Como este es el tiempo máximo de evento registrado hasta ahora por Spark, el umbral de la marca de agua se ha establecido 10 minutos antes (es decir, watermark_event_time es 16:52 h). Cualquier evento de datos con event_time posterior a las 16:52 h se tendrá en cuenta para la agregación con un límite de tiempo y cualquier evento de datos anterior a esa fecha se descartará. Esto permite a Spark mantener un estado intermedio durante 10 minutos adicionales para adaptarse a los datos tardíos. Alrededor de las 17:08 h, Spark recibió un evento con event_time a las 16:54 h, lo cual estaba dentro del umbral. Por lo tanto, Spark recalculó el intervalo entre las 16:50 y las 17:00, event-time-window y el volumen total pasó de 30 a 60.

Sin embargo, en la hora de activación de las 17:20 h, cuando Spark recibió un evento con event_time a las 17:15 h, se estableció watermark_event_time a las 17:05 h. Por lo tanto, el evento de datos tardío con event_time a las 17:03 h se consideró “demasiado tarde” y se ignoró.

```
Watermark Boundary = Max(Event Time) - Watermark Threshold
```

Uso de marcas de agua en AWS Glue

Spark no emitirá ni escribirá los datos en el receptor externo hasta que se supere el límite de la marca de agua. Para implementar una marca de agua en AWS Glue, consulte el siguiente ejemplo.

```
grouped_df = parsed_df \
                .withWatermark("event_time", "10 minutes") \
                .groupBy(window("event_time", "5 minutes"), "ticker") \
                .agg(sum("volume").alias("total_volume"))
```

Supervisión de los trabajos de transmisión de AWS Glue

Supervisar el trabajo de transmisión es una parte fundamental de la creación de su canalización de ETL. Además de usar la interfaz de usuario de Spark, también puede usar Amazon CloudWatch para supervisar las métricas. A continuación se muestra una lista de las métricas de transmisión que emite el marco de AWS Glue. Para obtener una lista completa de todas las métricas de AWS Glue, consulte Supervisión de AWS Glue mediante métricas de Amazon CloudWatch.

AWS Glue usa un marco de transmisión estructurado para procesar los eventos de entrada. Puede usar la API de Spark directamente en el código o aprovechar `ForEachBatch` que proporciona `GlueContext`, que publica estas métricas. Para entender estas métricas, primero debemos entender `windowSize`.

windowSize: `windowSize` es el intervalo de microlotes que usted proporciona. Si especifica un tamaño de ventana de 60 segundos, el trabajo de transmisión de AWS Glue esperará 60 segundos (o más si el lote anterior no se ha completado para entonces) antes de leer los datos de un lote procedentes del origen de transmisión y aplicar las transformaciones incluidas en `ForEachBatch`. Esto también se denomina intervalo de activación.

Revisemos las métricas con más detalle para comprender las características de estado y rendimiento.

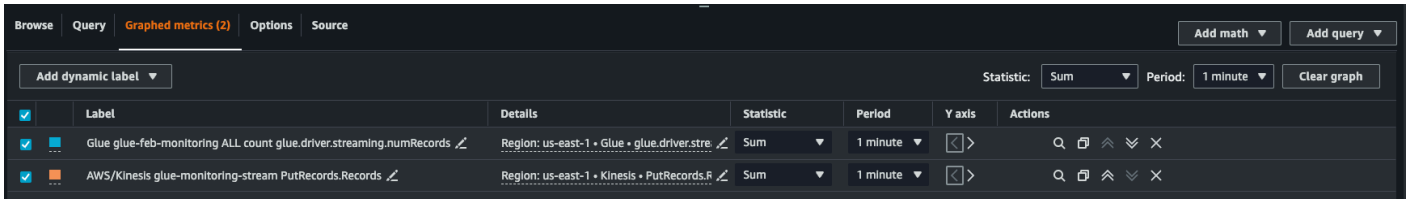
> **Note**
>
> Estas métricas se emiten cada 30 segundos. Si `windowSize` tiene menos de 30 segundos, las métricas informadas son una agregación. Por ejemplo, supongamos que `windowSize` tiene 10 segundos y que está procesando 20 registros de forma constante por microlote. En este escenario, el valor métrico emitido de numRecords sería 60.
> No se emite una métrica si no hay datos disponibles para ella. Además, en el caso de la métrica de retraso de consumo, debe habilitar la característica para obtener las métricas correspondientes.

Visualización de métricas

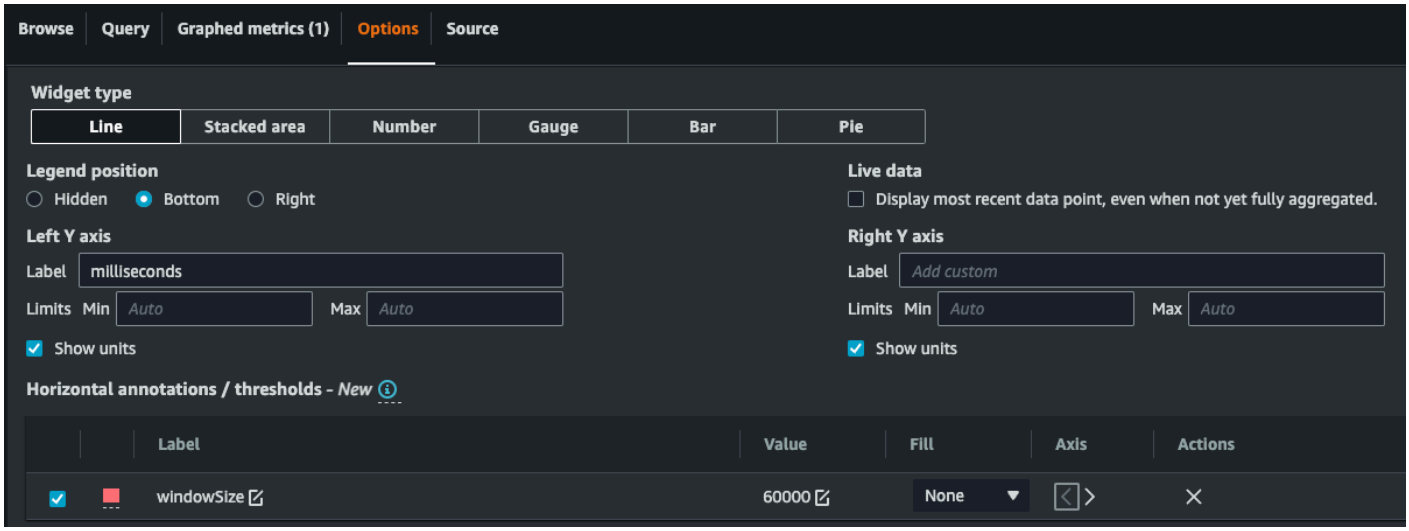
Para trazar métricas visuales:

1. Vaya a Métricas en la consola de Amazon CloudWatch y, a continuación, seleccione la pestaña Examinar. Luego elija Glue en “Espacios de nombres personalizados”.

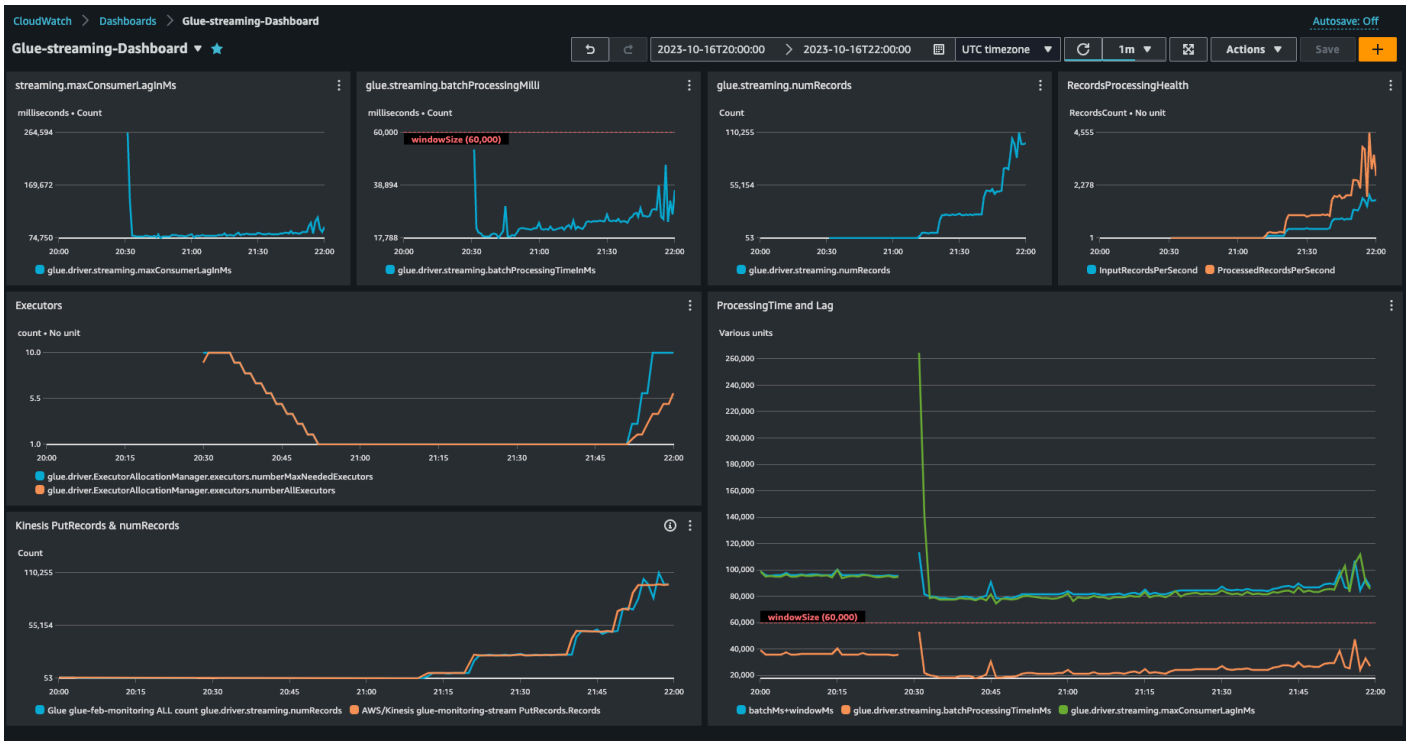
2. Seleccione Métricas de trabajo para ver las métricas de todos sus trabajos.
3. Filtre las métricas en función de JobName=glue-feb-monitoring y, a continuación, JobRunId=ALL. Puede hacer clic en el signo “+”, como se muestra en la siguiente figura, para agregarlo al filtro de búsqueda.
4. Seleccione la casilla de verificación de las métricas que le interesen. En la siguiente figura, hemos seleccionado `numberAllExecutors` y `numberMaxNeededExecutors`.



5. Una vez que haya seleccionado estas métricas, puede ir a la pestaña Métricas diagramadas y aplicarlas.
6. Como las métricas se emiten cada minuto, puede aplicar el “promedio” a lo largo de un minuto para obtener `batchProcessingTimeInMs` y `maxConsumerLagInMs`. Para `numRecords`, puede aplicar la “suma” a cada minuto.
7. Puede agegar una anotación horizontal de `windowSize` a su gráfico mediante la pestaña Opciones.



8. Una vez que haya seleccionado las métricas, cree un panel y agréguelo. A continuación, se ofrece un panel de ejemplo.

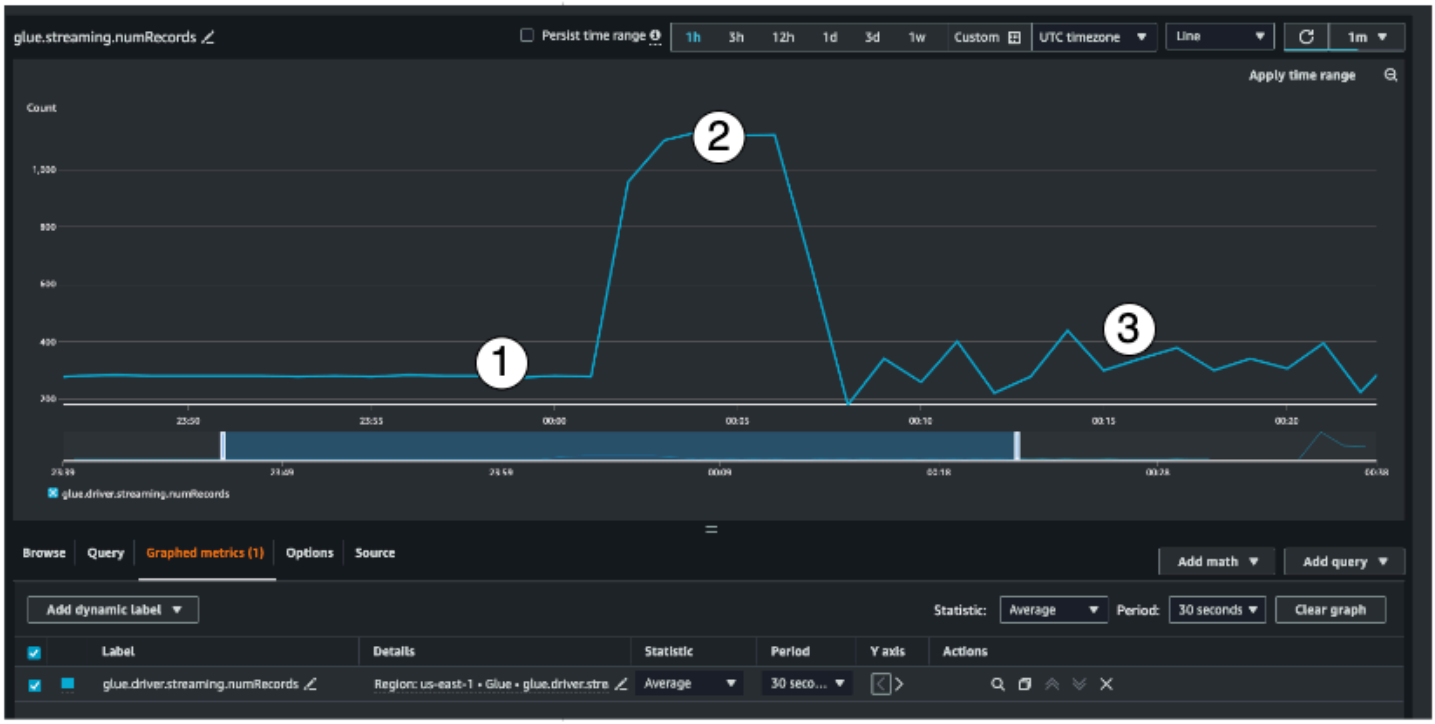


Análisis profundo de las métricas

En esta sección, se describe cada una de las métricas y cómo se relacionan entre sí.

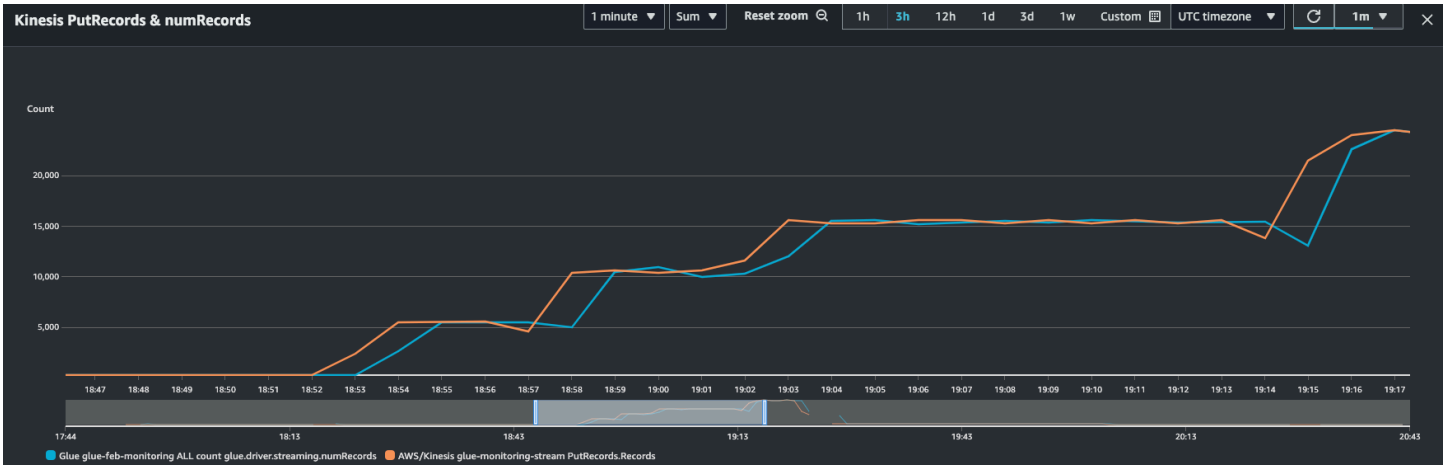
Número de registros (métrica: streaming.numRecords)

Esta métrica indica cuántos registros se están procesando.



Esta métrica de transmisión proporciona la visibilidad del número de registros que se están procesando en una ventana. Además de la cantidad de registros que se están procesando, también lo ayudará a comprender el comportamiento del tráfico de entrada.

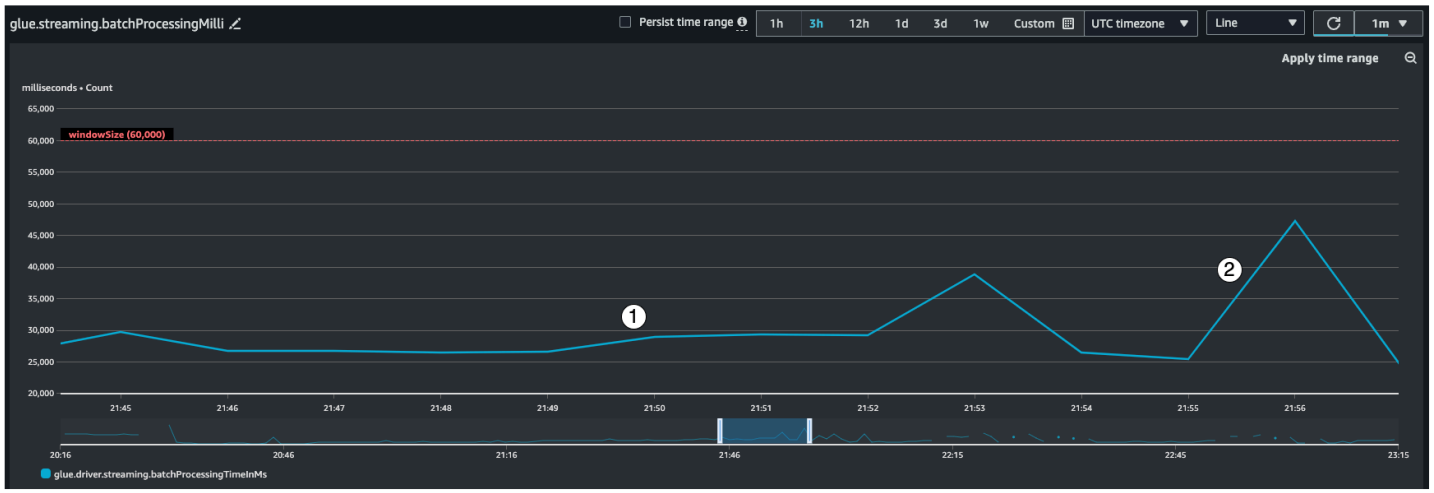
- El indicador n.º 1 es un ejemplo de un tráfico estable sin ráfagas. Por lo general, se trata de aplicaciones como los sensores de IoT que recopilan datos en intervalos regulares y los envían al origen de transmisión.
- El indicador n.º 2 es un ejemplo de una ráfaga repentina de tráfico en una carga que, por lo demás, es estable. Esto podría ocurrir en una aplicación de secuencias de clics cuando se lleva a cabo un evento de marketing, como el Black Friday, y se produce un aumento en el número de clics.
- El indicador n.º 3 es un ejemplo de tráfico impredecible. El tráfico impredecible significa que hay un problema. Es simplemente la naturaleza de los datos de entrada. Volviendo al ejemplo de los sensores de IoT, puede pensar en cientos de sensores que envían eventos de cambio climático al origen de transmisión. Como el cambio climático no es predecible, tampoco lo son los datos. Comprender el patrón de tráfico es clave para ajustar el tamaño de los ejecutores. Si la entrada tiene muchos picos, puede considerar usar el escalado automático (hablaremos de esto más adelante).



Puede combinar esta métrica con la métrica PutRecords de Kinesis para asegurarse de que el número de eventos que se ingiere y el número de registros que se leen son prácticamente iguales. Esto es útil en especial cuando se trata de entender el retraso. A medida que aumenta la tasa de ingesta, también lo hace la lectura de `numRecords` por parte de AWS Glue.

Tiempo de procesamiento por lotes (métrica: streaming.batchProcessingTimeInMs)

La métrica del tiempo de procesamiento por lotes lo ayuda a determinar si el clúster está infra o sobreaprovisionado.

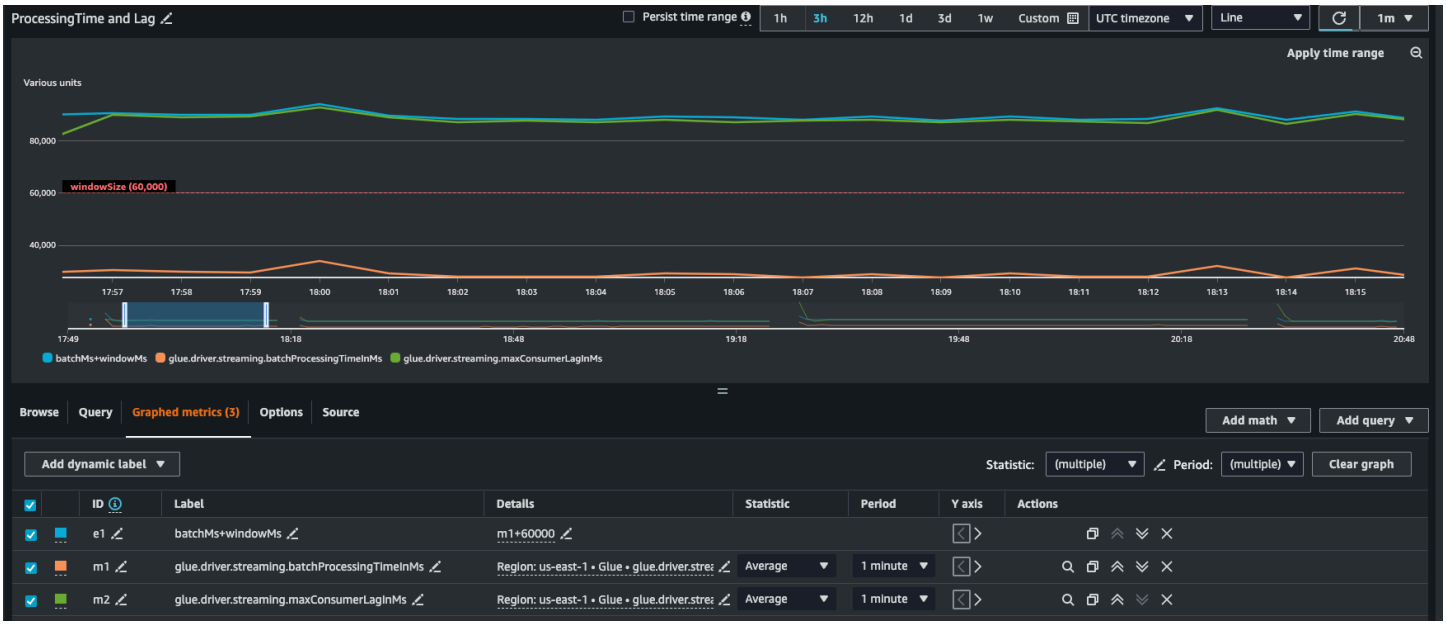


Esta métrica indica el número de milisegundos que se tardó en procesar cada microlote de registros. El objetivo principal aquí es supervisar este tiempo para asegurarse de que sea inferior al intervalo de `windowSize`. No hay problema si `batchProcessingTimeInMs` se suspende temporalmente, siempre y cuando se recupere en el siguiente intervalo. El indicador n.º 1 representa un tiempo más o menos estable que se tarda en procesar el trabajo. Sin embargo, si el número de registros de entrada aumenta, el tiempo necesario para procesar el trabajo aumentará, tal como se muestra en

el indicador n.º 2. Si `numRecords` no aumenta, pero el tiempo de procesamiento sí, entonces tendrá que analizar más a fondo el procesamiento de los trabajos en los ejecutores. Es una buena práctica establecer un umbral y una alarma para garantizar que `batchProcessingTimeInMs` no supere el 120 % durante más de 10 minutos. Para obtener más información acerca de la configuración de alarmas, consulte Uso de alarmas de Amazon CloudWatch.

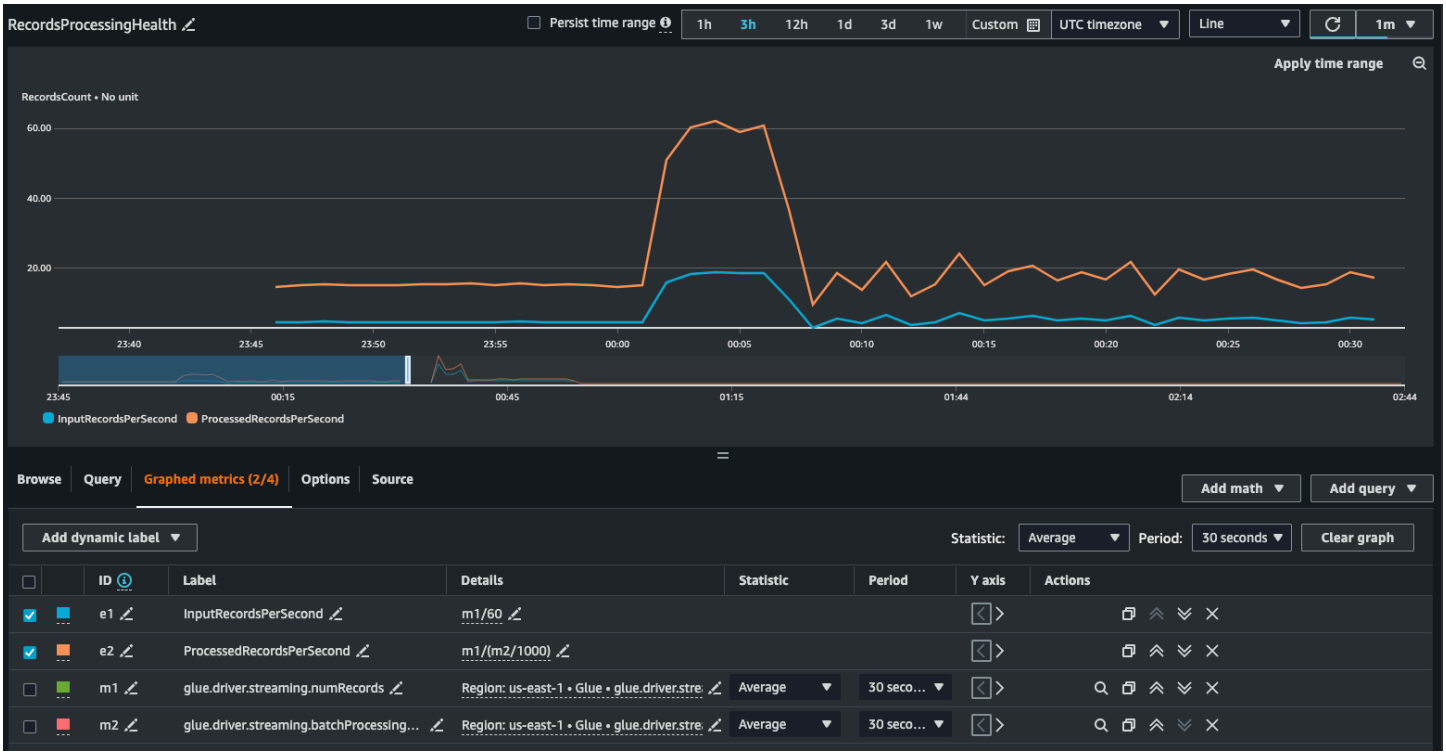
Retraso de consumo (métrica: streaming.maxConsumerLagInMs)

La métrica de retraso de consumo lo ayuda a comprender si hay un retraso en el procesamiento de los eventos. Si el retraso es demasiado alto, es posible que no cumpla con el SLA de procesamiento del que depende su empresa, aunque tenga un windowSize correcto. Tiene que habilitar estas métricas de forma explícita mediante la opción de conexión de `emitConsumerLagMetrics`. Para obtener más información, consulte KinesisStreamingSourceOptions.



Métricas derivadas

Para obtener información más detallada, puede crear métricas derivadas para comprender mejor sus trabajos de transmisión en Amazon CloudWatch.



Puede crear un gráfico con métricas derivadas para decidir si necesita usar más DPU. Si bien el escalado automático lo ayuda a hacerlo automáticamente, puede usar métricas derivadas para determinar si el escalado automático funciona de manera eficaz.

- InputRecordsPerSecond indica la velocidad a la que se obtienen los registros de entrada. Se deriva de la siguiente manera: número de registros de entrada (glue.driver.streaming.numRecords)/ WindowSize.
- ProcessingRecordsPerSecond indica la velocidad a la que se procesan los registros. Se deriva de la siguiente manera: número de registros de entrada (glue.driver.streaming.numRecords)/ batchProcessingTimeInMs.

Si la velocidad de entrada es superior a la velocidad de procesamiento, es posible que necesite agregar más capacidad para procesar el trabajo o aumentar el paralelismo.

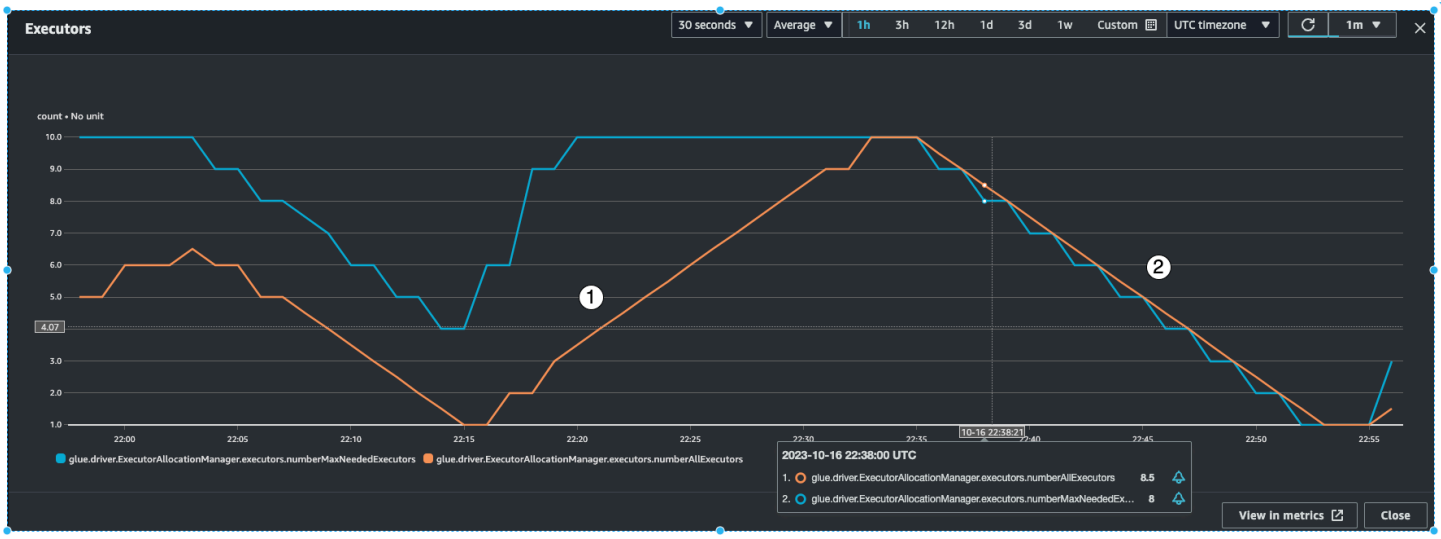
Métricas de escalado automático

Si el tráfico de entrada tiene picos, debería considerar la posibilidad de habilitar el escalado automático y especificar el número máximo de trabajos. Con eso, obtiene dos métricas adicionales: `numberAllExecutors` y `numberMaxNeededExecutors`.

- numberAllExecutors es el número de ejecutores de trabajo que se ejecutan activamente.

- numberMaxNeededExecutors es el número máximo de ejecutores de trabajos (en ejecución activa y pendientes) necesarios para satisfacer la carga actual.

Estas dos métricas lo ayudarán a entender si el escalado automático funciona correctamente.



AWS Glue supervisará la métrica `batchProcessingTimeInMs` en unos pocos microlotes y hará una de las siguientes dos cosas: Si `batchProcessingTimeInMs` está más cerca de `windowSize`, escalará horizontalmente los ejecutores; si `batchProcessingTimeInMs` es inferior a `windowSize`, reducirá horizontalmente los ejecutores. Además, usará un algoritmo para escalar de forma escalonada los ejecutores.

- El indicador n.º 1 representa cómo los ejecutores activos se escalaron verticalmente a fin de alcanzar el máximo número de ejecutores necesarios para procesar la carga.
- El indicador n.º 2 representa cómo se redujeron horizontalmente los ejecutores activos debido a que `batchProcessingTimeInMs` era inferior.

Puede usar estas métricas para supervisar el paralelismo actual a nivel de ejecutor y ajustar el número máximo de trabajos en la configuración de escalado automático en consecuencia.

Cómo obtener el mejor rendimiento

Spark intentará crear una tarea por fragmento para poder leerla en el flujo de Amazon Kinesis. Los datos de cada fragmento se convierten en una partición. Luego, distribuirá estas tareas entre los ejecutores o trabajos, en función del número de núcleos de cada trabajo (el número de núcleos por trabajo depende del tipo de trabajo que seleccione, como `G.025X`, `G.1X`, etc.). Sin embargo, la

forma en que se distribuyen las tareas no es determinista. Todas las tareas se ejecutan en paralelo en sus respectivos núcleos. Si hay más fragmentos que el número de núcleos ejecutores disponibles, las tareas se ponen en cola.

Puede usar una combinación de las métricas anteriores y el número de fragmentos para aprovisionar los ejecutores a fin de garantizar una carga estable con espacio para las ráfagas. Se recomienda realizar algunas iteraciones del trabajo para determinar el número aproximado de trabajos. Para una carga de trabajo inestable o con picos, puede hacer lo mismo al configurar el escalado automático y el número máximo de trabajos.

Configure `windowSize` de acuerdo con los requisitos de SLA de su empresa. Por ejemplo, si su empresa exige que los datos procesados no puedan estar inactivos durante más de 120 segundos, configure `windowSize` en al menos 60 segundos, de forma que el retraso promedio de consumo sea inferior a 120 segundos (consulte la sección anterior acerca del retraso de consumo). A partir de ahí, en función de `numRecords` y del número de fragmentos, planifique la capacidad de las DPU asegurándose de que `batchProcessingTimeInMs` sea inferior al 70 % de `windowSize` la mayor parte del tiempo.

> **Note**
>
> Los fragmentos activos pueden distorsionar los datos, lo que significa que algunos fragmentos o particiones son mucho más grandes que otros. Esto puede provocar que algunas tareas que se ejecutan en paralelo tarden más tiempo, lo que genera que las tareas queden rezagadas. Como resultado, el siguiente lote no puede comenzar hasta que se hayan completado todas las tareas del anterior, lo que afectará `batchProcessingTimeInMillis` y el retraso máximo.

AWS Glue Calidad de los datos

AWS Glue La calidad de los datos le permite medir y supervisar la calidad de los datos para poder tomar buenas decisiones empresariales. Basado en un DeeQu marco de código abierto, AWS Glue Data Quality proporciona una experiencia gestionada y sin servidores. AWS Glue Data Quality funciona con el lenguaje de definición de calidad de datos (DQDL), que es un lenguaje de dominio específico que se utiliza para definir las normas de calidad de los datos. Para obtener más información sobre el DQDL y los tipos de reglas compatibles, consulte Referencia del lenguaje de definición de calidad de datos (DQDL).

Para conocer los detalles adicionales del producto y los precios, consulte la página de servicio de Calidad de datos de AWS Glue.

Beneficios y características principales

Los beneficios y las características clave de la calidad de AWS Glue los datos incluyen:

- Sin servidor: no requiere instalación, aplicación de parches ni mantenimiento.
- Comience rápidamente: AWS Glue Data Quality analiza rápidamente sus datos y crea reglas de calidad de datos para usted. Puede empezar con dos clics: “Crear reglas de calidad de datos → Recomendar reglas”.
- Detecte problemas de calidad de los datos: utilice el aprendizaje automático (ML) para detectar anomalías y problemas de calidad de hard-to-detect los datos.
- Improvise sus reglas: con más de 25 reglas de out-of-the-box DQ para empezar, puede crear reglas que se adapten a sus necesidades específicas.
- Evalúe la calidad y tome decisiones empresariales con confianza: una vez que evalúe las reglas, obtendrá una puntuación de calidad de los datos que proporciona una visión general del estado de sus datos. Utilice la puntuación de calidad de los datos para tomar decisiones empresariales fiables.
- Céntrese en los datos incorrectos: la calidad de AWS Glue los datos le ayuda a identificar los registros exactos que provocaron la caída de sus puntajes de calidad. Identifíquelos fácilmente, póngalos en cuarentena y corríjalos.
- Pague por uso: no necesita licencias anuales para utilizar AWS Glue Data Quality.
- Sin restricciones: AWS Glue Data Quality se basa en el código abierto DeeQu, lo que le permite mantener las reglas que está creando en un lenguaje abierto.

- Controles de calidad de AWS Glue los datos: calidad de los datos Puede aplicar controles de calidad de los datos Data Catalog y los procesos de AWS Glue ETL, lo que le permitirá gestionar la calidad de los datos en reposo y en tránsito.
- Detección de la calidad de los datos basada en el aprendizaje automático: utilice el aprendizaje automático (ML) para detectar anomalías y hard-to-detect problemas de calidad de los datos.

Funcionamiento

Hay dos puntos de partida para la calidad de AWS Glue los datos: los trabajos AWS Glue Data Catalog y AWS Glue los de ETL. Esta sección proporciona una descripción general de los casos de uso y AWS Glue las características que admite cada punto de entrada.

Calidad de los datos para AWS Glue Data Catalog

AWS Glue La calidad de los datos evalúa los objetos que se almacenan en el. AWS Glue Data Catalog Ofrece a los no codificadores una forma sencilla de configurar las reglas de calidad de los datos. Entre estas personas, se incluyen administradores de datos y analistas de negocios.

Puede elegir esta opción para los siguientes casos de uso:

- Desea realizar tareas de calidad de datos en conjuntos de datos que ya ha catalogado en AWS Glue Data Catalog.
- Trabaja en la gobernanza de datos y necesita identificar o evaluar los problemas de calidad de los datos en su lago de datos de forma continua.

Puede administrar la calidad de los datos del catálogo de datos mediante las siguientes interfaces:

- La consola de administración AWS Glue
- AWS Glue APIs

Para empezar con AWS Glue Data Quality for the, AWS Glue Data Catalog consulteIntroducción a AWS Glue Data Quality para el Data Catalog.

Calidad de datos para trabajos de AWS Glue ETL

AWS Glue La calidad de los datos para los trabajos de AWS Glue ETL le permite realizar tareas proactivas de calidad de datos. Las tareas proactivas ayudan a identificar y filtrar los datos incorrectos antes de cargar un conjunto de datos en el lago de datos.

Vídeo: Presentamos la calidad AWS Glue de los datos para ETL Pipelines

Puede elegir la calidad de los datos para los trabajos de ETL en los siguientes casos de uso:

- Desea incorporar tareas de calidad de datos en sus trabajos de ETL
- Desea escribir código que defina las tareas de calidad de los datos en los scripts de ETL
- Desea administrar la calidad de los datos que fluyen en los procesos de datos visuales

Puede administrar la calidad de los datos para los trabajos de ETL mediante las siguientes interfaces:

- AWS Glue Studio, AWS Glue Studio cuadernos y sesiones interactivas AWS Glue
- AWS Glue bibliotecas para secuencias de comandos ETL
- AWS Glue API

Para empezar con la calidad de los datos para los trabajos de ETL, consulte Tutorial: Introducción a Calidad de datos en la Guía del usuario de AWS Glue Studio .

Comparación de la calidad de los datos del catálogo de datos con la calidad de los datos para los trabajos de ETL

Esta tabla proporciona una descripción general de las funciones que admite cada punto de entrada para la calidad de AWS Glue los datos.

Característica	Calidad de datos para el catálogo de datos	Calidad de los datos para los trabajos de ETL
Origen de datos	Orígenes de Amazon S3, Amazon Redshift y JDBC compatibles con el catálogo de datos y formatos de lago	Todas las fuentes de datos son compatibles AWS Glue, incluidos los conectore

Característica	Calidad de datos para el catálogo de datos	Calidad de los datos para los trabajos de ETL
	de datos transaccionales, como Apache Iceberg, Apache Hudi y Delta Lake. Tenga en cuenta que si se AWS Lake Formation administr an tablas, no se admiten las tablas Iceberg, Delta y HUDI. Amazon Athena no se admiten las vistas AWS Glue Data Catalog catalogadas.	s personalizados y los conectores de terceros.
Recomendaciones sobre reglas de calidad de datos	Compatible	No compatible
Crear y ejecutar reglas de DQDL	Soportado	Soportado
Escalado automático	No compatible	Compatible
AWS Glue Soporte flexible	No compatible	Compatible
Programación	Compatible al evaluar las reglas de calidad de los datos y mediante Step Functions.	Compatible al usar Step Functions y flujos de trabajo.
Identificación de registros que no superaron las comprobac iones de calidad de los datos	No compatible	Compatible
Integración con Amazon Eventbridge	Soportado	Soportado
Integración con AWS Cloudwatch	Soportado	Soportado

Característica	Calidad de datos para el catálogo de datos	Calidad de los datos para los trabajos de ETL
Escritura de resultados de calidad de datos en Amazon S3	Soportado	Soportado
Calidad de datos incremental	Compatible mediante predicados insertados	Compatible a través de marcadores AWS Glue
AWS CloudFormation soporte	Soportado	Soportado
Detección de anomalías basada en machine learning	No compatible	Vista previa
Reglas dinámicas	No compatible	Compatible

Consideraciones

Tenga en cuenta lo siguiente antes de utilizar AWS Glue Data Quality:

- Las reglas de calidad de datos no pueden evaluar los orígenes de datos anidados o de tipo lista. Consulte Aplanamiento de estructuras anidadas.

Terminología

La siguiente lista define los términos relacionados con la calidad de AWS Glue los datos.

Lenguaje de definición de calidad de datos (DQDL)

Lenguaje específico de un dominio que puede usar para escribir reglas de calidad AWS Glue de datos.

Para obtener más información sobre DQDL, consulte la guía de Referencia del lenguaje de definición de calidad de datos (DQDL).

calidad de datos

Describe en qué medida un conjunto de datos cumple su propósito específico. AWS Glue La calidad de los datos evalúa las reglas comparándolas con un conjunto de datos para medir la calidad de los datos. Cada regla comprueba características específicas, como la actualización o integridad de los datos. Para cuantificar la calidad de los datos, puede utilizar una puntuación de calidad de datos.

puntuación de calidad de datos

El porcentaje de reglas de calidad de datos que se aprueban (el resultado es verdadero) al evaluar un conjunto de reglas con AWS Glue Data Quality.

regla

Una expresión de DQDL que comprueba los datos para detectar una característica específica y devuelve un valor booleano. Para obtener más información, consulte Estructura de la regla.

analizador

Expresión DQDL que recopila estadísticas de datos. Un analizador recopila estadísticas de datos que los algoritmos de aprendizaje automático pueden utilizar para detectar anomalías y hard-to-detect problemas de calidad de los datos a lo largo del tiempo.

conjunto de reglas

AWS Glue Recurso que comprende un conjunto de reglas de calidad de los datos. El conjunto de reglas debe estar asociado a una tabla de AWS Glue Data Catalog. Al guardar un conjunto de reglas, AWS Glue asigna un nombre de recurso de Amazon (ARN) al conjunto de reglas.

puntuación de calidad de datos

El porcentaje de reglas de calidad de datos que cumplen (el resultado es true [verdadero]) al evaluar un conjunto de reglas con Calidad de datos de AWS Glue.

observación

Información no confirmada que se genera por AWS Glue cuando se analizan las estadísticas de datos recopiladas por las reglas y los analizadores a lo largo del tiempo.

Límites

AWS Glue Límites del servicio de calidad de los datos:

- Puede tener 2000 reglas en un conjunto de reglas. Si sus conjuntos de reglas son más grandes, le recomendamos dividirlos en varios conjuntos de reglas.
- El tamaño del conjunto de reglas es de 65 KB. Si sus conjuntos de reglas son más grandes, le recomendamos dividirlos en varios conjuntos de reglas.

Notas de publicación sobre la calidad de los datos AWS Glue

En este tema se describen las funciones introducidas en AWS Glue Data Quality.

Disponibilidad general: características nuevas

Las siguientes funciones nuevas están disponibles con la disponibilidad general de AWS Glue Data Quality:

- La capacidad de identificar qué registros no superaron las comprobaciones de calidad de los datos ahora se admite en AWS Glue Studio
- Nuevos tipos de reglas de calidad de los datos, como la validación de la integridad referencial de los datos entre dos conjuntos de datos, la comparación de datos entre dos conjuntos de datos y las comprobaciones del tipo de datos
- Experiencia de usuario mejorada en AWS Glue Data Catalog
- Compatibilidad con Apache Iceberg, Apache Hudi y Delta Lake
- Compatibilidad con Amazon Redshift
- Notificación simplificada con Amazon EventBridge
- AWS CloudFormation soporte para crear conjuntos de reglas
- Mejoras en el rendimiento: opción de almacenamiento en caché en ETL y AWS Glue Studio para un rendimiento más rápido al evaluar la calidad de los datos

27 de noviembre de 2023 (Vista previa)

- Las capacidades de detección de anomalías impulsadas por ML ahora están disponibles en AWS Glue ETL y AWS Glue Studio. Con esto, ahora puede detectar anomalías y problemas de calidad de los hard-to-detect datos.
- Las reglas dinámicas le permiten dar umbrales dinámicos (por ejemplo: `RowCount> avg(last(10))`).

12 de marzo de 2024

- Mejoras del DQDL
  - Compatibilidad con palabras clave como NULL, BLANKS y WHITESPACES_ONLY
  - Opciones para especificar cómo debe gestionar AWS Glue Data Quality las reglas compuestas
  - ColumnValues el tipo de regla no permitirá que se pasen valores NULOS durante las comparaciones
  - Compatibilidad con el operador NOT en el DQDL

26 de junio de 2024

- Mejoras del DQDL
  - El DQDL ahora admite la cláusula where para que pueda filtrar los datos antes de aplicar las reglas DQ

Detección de anomalías en Calidad de los datos AWS Glue

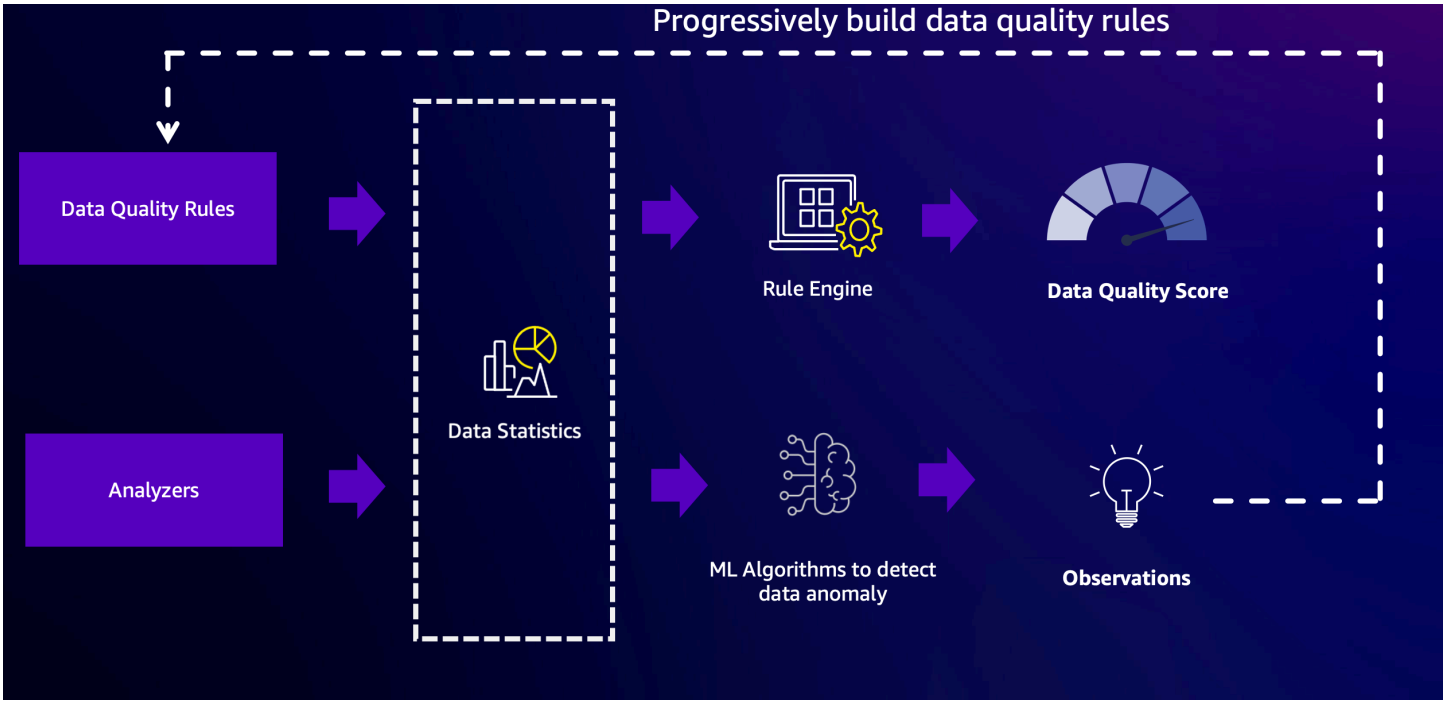
> **Note**
>
> La calidad de datos de AWS Glue está disponible en las siguientes regiones:
>
> - Este de EE. UU. (Ohio, Norte de Virginia)
> - Oeste de EE. UU. (Oregón)
> - Asia-Pacífico (Tokio)
> - Europa (Irlanda)

La detección de anomalías en la calidad de los datos de AWS Glue aplica algoritmos de machine learning (ML) a las estadísticas de datos a lo largo del tiempo para detectar patrones anormales y problemas ocultos de calidad de los datos que son difíciles de detectar con reglas. En la actualidad, la detección de anomalías solo está disponible en la versión 4.0. de AWS Glue. Actualmente, esta función solo está disponible en AWS Glue Studio Visual ETL y AWS Glue ETL. Esta capacidad no funciona en los blocs de notas de AWS Glue Studio, el catálogo de datos de AWS Glue, las sesiones interactivas de AWS Glue y las vistas previas de datos de AWS Glue.

Funcionamiento

Al evaluar las reglas de calidad de los datos, AWS Glue captura las estadísticas de datos necesarias para determinar si los datos cumplen las reglas. Por ejemplo, Calidad de los datos calculará el número de valores distintos de un conjunto de datos y, a continuación, comparará ese valor con las expectativas.

El motor de reglas de Calidad de los datos compara el valor estadístico con los umbrales definidos y evalúa los requisitos de calidad. Según estas estadísticas se recopilan a lo largo del tiempo, puede habilitar la detección de anomalías en sus canalizaciones de ETL para permitir que AWS Glue aprenda de las estadísticas anteriores e informe de los patrones ocultos como observaciones. Las observaciones son información no confirmada que identifica el algoritmo de machine learning de AWS Glue. Incluyen reglas de calidad de los datos recomendadas que puede aplicar a su conjunto de reglas para monitorear el patrón descubierto. Recomendamos ejecutar los trabajos en un horario habitual (por ejemplo, cada hora y todos los días). Las ejecuciones irregulares pueden producir una mala percepción.



Uso de analizadores para inspeccionar los datos

A veces, es posible que no tenga tiempo para crear reglas de calidad de los datos. En estos casos es donde los analizadores resultan útiles. Los analizadores forman parte de su conjunto de reglas y son muy fáciles de configurar. Por ejemplo, puede escribir lo siguiente en su conjunto de reglas:

```
Analyzers = [
      RowCount,
      Completeness “AllColumns”
]
```

Esto recopilará las siguientes estadísticas:

- Recuento de filas para todo el conjunto de datos
- Integridad de cada columna en su conjunto de datos

Recomendamos usar analizadores porque así no tendrá que preocuparse por los umbrales. Puede ejecutar sus canalizaciones de datos y, tras tres ejecuciones, Calidad de los datos de AWS Glue empezará a generar observaciones y recomendaciones de reglas cuando detecte cualquier anomalía. Puede revisar las observaciones y las estadísticas asociadas e incorporar fácilmente las recomendaciones de reglas en su conjunto de reglas. Para empezar, consulte Configuración de la detección de anomalías y generación de información . Tenga en cuenta que los analizadores no afectarán a las puntuaciones de calidad de sus datos. Generan estadísticas que se pueden analizar a lo largo del tiempo para generar observaciones.

Uso de la DetectAnomaly regla

A veces, uno quiere que su trabajo falle cuando al momento de detectar anomalías. Para hacer cumplir una restricción, debe configurar una regla. Los analizadores no detendrán un trabajo. En su lugar, recopilarán estadísticas y analizarán los datos. Al configurar la regla `DetectAnomaly` en la sección de reglas del conjunto de reglas, se confirmará que el escaneo DQ informa que el trabajo no ha superado todas las reglas del escaneo.

Ventajas y casos de uso de la detección de anomalías

Los ingenieros pueden gestionar cientos de canalizaciones de datos en un momento dado. Cada canalización puede extraer datos de diferentes fuentes y cargarlos en el lago de datos. Dado que cada canalización puede extraer datos de una fuente diferente y cargarlos en un lago de datos, es difícil obtener información inmediata sobre los datos, ya sea que su forma haya cambiado significativamente o que se alejara de las tendencias existentes.

En el pasado, las fuentes de datos preliminares cambiaban sin previo aviso a los equipos de ingeniería de datos, lo que hard-to-track introducía «errores de datos» en este proceso. Añadir nodos

de calidad de los datos a los trabajos facilita mucho las cosas, ya que los trabajos fallan cuando se detectan problemas. Sin embargo, esto no elimina todos los modos de fallo que preocupan a los equipos de datos, lo que deja la puerta abierta a que aparezcan otros errores de datos.

Uno de los modos de fallo está relacionado con el volumen de datos. Conforme el almacén de datos de una empresa crece con el tiempo, la cantidad de registros producidos por las canalizaciones de datos puede aumentar exponencialmente. Cada semana, es posible que los equipos de datos tengan que actualizar manualmente los trabajos de ETL para aumentar cada regla de calidad de los datos que establece un límite al número de filas incorporadas.

Otro error es que algunos de los límites de las normas de calidad de los datos son muy amplios para tener en cuenta el hecho de que el volumen de transacciones varía dependiendo del día de la semana. Los fines de semana casi no hay transacciones, y los lunes hay aproximadamente tres veces más transacciones que otros días. Los equipos de datos tienen dos opciones: implementar la lógica para cambiar el conjunto de reglas sobre la marcha dependiendo del día o establecer una expectativa muy amplia.

Por último, a los equipos de datos también les preocupan los errores de datos menos definidos con claridad. Los modelos se han entrenado con datos con características específicas, y si estos comienzan a sesgarse de forma inesperada, el equipo querrá saberlo. Por ejemplo, en febrero, una empresa puede expandirse a Montana y, por lo tanto, las transacciones que empezaron a contener el código “MT” aparecen con más frecuencia. Esto pudo haber roto la inferencia del machine learning y, como resultado, los modelos predijeron falsamente que todas las transacciones de Montana eran fraudulentas.

Aquí es donde la detección de anomalías en la calidad de los datos puede ayudar a resolver estos problemas. Algunos de los beneficios de la detección de anomalías en la calidad de los datos incluyen:

- Escaneo de datos de forma programada, basada en eventos o manual.
- Detección de anomalías que pueden ser indicativas de un evento imprevisto, de una estacionalidad o de una anomalía estadística.
- Ofrezca recomendaciones de reglas para tomar medidas en relación con las observaciones detectadas mediante la detección de anomalías en la calidad de los datos.

Esto resulta útil si:

- quiere detectar anomalías en sus datos automáticamente sin necesidad de escribir reglas de calidad de los datos.
- quiere atrapar posibles problemas en sus datos que las reglas de calidad de los datos por sí solas no pueden encontrar.
- quiere automatizar algunas tareas que evolucionan con el tiempo, como limitar el número de filas utilizadas para supervisar la calidad de los datos.

Permisos de IAM para Calidad de datos de AWS Glue

En este tema se proporciona información para comprender las acciones y los recursos que usted, un administrador de IAM, puede utilizar en una política (de IAM) de AWS Identity and Access Management para Calidad de datos de AWS Glue. También incluye ejemplos de políticas de IAM con los permisos mínimos que necesita para usar Calidad de los datos de AWS Glue con el catálogo de datos de Glue de AWS.

Para obtener más información sobre la seguridad en Glue de AWS, consulte Seguridad en AWS Glue.

Permisos de IAM para Calidad de los datos de AWS Glue

En la siguiente tabla, se enumeran los permisos necesita un usuario para poder llevar a cabo operaciones específicas de la Calidad de datos de AWS Glue. Para establecer una autorización detallada para Calidad de datos de AWS Glue, puede especificar estas acciones en el elemento `Action` de una instrucción de política de IAM.

Acciones de Calidad de datos de AWS Glue

Acción	Descripción	Tipos de recurso
`glue:CreateDataQualityRuleset`	Concede permiso para crear un conjunto de reglas de calidad de datos.	`::dataQualityRuleset/<name>`
`glue:DeleteDataQualityRuleset`	Concede permiso para eliminar un conjunto de reglas de calidad de datos.	`::dataQualityRuleset/<name>`

Acción	Descripción	Tipos de recurso
glue:GetDataQualityRuleset	Concede permiso para recuperar un conjunto de reglas de calidad de datos.	::dataQualityRuleset/<name>
glue:ListDataQualityRulesets	Concede permiso para recuperar todos los conjuntos de reglas de calidad de datos.	::dataQualityRuleset/*
glue:UpdateDataQualityRuleset	Concede permiso para actualizar un conjunto de reglas de calidad de datos.	::dataQualityRuleset/<name>
glue:GetDataQualityResult	Concede permiso para recuperar un resultado de la ejecución de tareas de calidad de datos.	::dataQualityRuleset/<name>
glue:ListDataQualityResults	Concede permiso para recuperar todos los resultados de la ejecución de tareas de calidad de datos.	::dataQualityRuleset/*
glue:CancelDataQualityRuleRecommendationRun	Concede permiso para detener una ejecución en curso de tareas de recomendación de calidad de datos.	::dataQualityRuleset/*
glue:GetDataQualityRuleRecommendationRun	Concede permiso para recuperar una ejecución de tareas de recomendación de calidad de datos.	::dataQualityRuleset/*
glue:ListDataQualityRuleRecommendationRuns	Concede permiso para recuperar todas las ejecuciones de tareas de recomendación de calidad de datos.	::dataQualityRuleset/*

Acción	Descripción	Tipos de recurso
glue:StartDataQualityRuleRecommendationRun	Concede permiso para iniciar una ejecución de tareas de recomendación de calidad de datos.	::dataQualityRuleset/*
glue:CancelDataQualityRulesetEvaluationRun	Concede permiso para detener una ejecución en curso de tareas de calidad de datos.	::dataQualityRuleset/*
glue:GetDataQualityRulesetEvaluationRun	Concede permiso para recuperar una ejecución de tareas de calidad de datos.	::dataQualityRuleset/*
glue:ListDataQualityRulesetEvaluationRuns	Concede permiso para recuperar todas las ejecuciones de tareas de calidad de datos.	::dataQualityRuleset/*
glue:StartDataQualityRulesetEvaluationRun	Concede permiso para iniciar una ejecución de tareas de calidad de datos.	::dataQualityRuleset/<name>
glue:PublishDataQuality	Concede permiso para publicar los resultados de calidad de datos	::dataQualityRuleset/<name>

Se requiere una configuración de IAM para programar las ejecuciones de evaluación

Permisos de IAM

Para realizar ejecuciones programadas de evaluación de Calidad de los datos, debe agregar la acción `IAM:PassRole` a la política de permisos.

Permisos necesarios para AWS EventBridge Scheduler

Acción	Descripción	Tipos de recurso
`iam:PassRole`	Otorga permiso a IAM para que el usuario pueda pasar las funciones aprobadas.	ARN del rol utilizado para llamar `StartDataQualityRulesetEvaluationRun`

Sin estos permisos, se produce el siguiente error:

```
"errorCode": "AccessDenied"
"errorMessage": "User: arn:aws:sts::account_id:assumed-role/AWSGlueServiceRole is not
authorized to perform: iam:PassRole on resource: arn:aws:iam::account_id:role/service-
role/AWSGlueServiceRole
because no identity-based policy allows the iam:PassRole action"
```

Entidades de confianza de IAM

Los servicios de AWS Glue y AWS EventBridge Scheduler deben figurar en las entidades de confianza para poder crear y ejecutar una `StartDataQualityEvaluationRun` programada.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "glue.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        },
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "scheduler.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
```

```
}
```

Ejemplos de políticas de IAM

Un rol de IAM para Calidad de los datos de AWS Glue necesita los siguientes tipos de permisos:

- Permisos para las operaciones de Calidad de los datos de AWS Glue, de modo que pueda obtener las reglas de calidad de datos recomendadas y ejecutar una tarea de calidad de datos en una tabla del Catálogo de datos de AWS Glue. Los ejemplos de políticas de IAM de esta sección incluyen los permisos mínimos necesarios para las operaciones de Calidad de los datos de AWS Glue.
- Permisos que otorgan acceso a la tabla del Catálogo de datos y a los datos subyacentes. Estos permisos varían en función de su caso de uso. Por ejemplo, para los datos que cataloga en Amazon S3, los permisos deben incluir el acceso a Amazon S3.

> **Note**
>
> Debe configurar los permisos de Amazon S3 además de los permisos descritos en esta sección.

Permisos mínimos para obtener las reglas de calidad de datos recomendadas

Esta política de ejemplo incluye los permisos que necesita para generar las reglas de calidad de datos recomendadas.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowGlueRuleRecommendationRunActions",
      "Effect": "Allow",
      "Action": [
        "glue:GetDataQualityRuleRecommendationRun",
        "glue:PublishDataQuality",
        "glue:CreateDataQualityRuleset"
      ],
      "Resource": "arn:aws:glue:us-east-1:111122223333:dataQualityRuleset/*"
    },
 {
     "Sid": "AllowCatalogPermissions",
```

```
     "Effect": "Allow",
     "Action": [
        "glue:GetPartitions",
        "glue:GetTable"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Sid": "AllowS3GetObjectToRunRuleRecommendationTask",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": "arn:aws:s3:::aws-glue-*"
    },
 {   // Optional for Logs
        "Sid": "AllowPublishingCloudwatchLogs",
        "Effect": "Allow",
        "Action": [
          "logs:CreateLogStream",
          "logs:CreateLogGroup",
          "logs:PutLogEvents"
        ],
        "Resource": "*"
    },
  ]
}
```

Permisos mínimos para ejecutar una tarea de calidad de datos

Esta política de ejemplo incluye los permisos que necesita para ejecutar una tarea de evaluación de calidad de los datos.

Las siguientes instrucciones de políticas son opcionales en función de su caso de uso:

- `AllowCloudWatchPutMetricDataToPublishTaskMetrics`: es obligatoria para publicar las métricas de ejecución de calidad de datos en Amazon CloudWatch.
- `AllowS3PutObjectToWriteTaskResults`: es obligatoria para escribir los resultados de ejecución de calidad de datos en Amazon S3.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowGlueGetDataQualityRuleset",
      "Effect": "Allow",
      "Action": [
        "glue:GetDataQualityRuleset"
      ],
      "Resource": "arn:aws:glue:us-east-1:111122223333:dataQualityRuleset/<YOUR-
RULESET-NAME>"
    },
    {
      "Sid": "AllowGlueRulesetEvaluationRunActions",
      "Effect": "Allow",
      "Action": [
        "glue:GetDataQualityRulesetEvaluationRun",
        "glue:PublishDataQuality"
      ],
      "Resource": "arn:aws:glue:us-east-1:111122223333:dataQualityRuleset/*"
    },
 {
      "Sid": "AllowCatalogPermissions",
      "Effect": "Allow",
      "Action": [
        "glue:GetPartitions",
        "glue:GetTable"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Sid": "AllowS3GetObjectForRulesetEvaluationRun",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": "arn:aws:s3:::aws-glue-*"
    },
    {
      "Sid": "AllowCloudWatchPutMetricDataToPublishTaskMetrics",
      "Effect": "Allow",
```

```
      "Action": [
        "cloudwatch:PutMetricData"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "cloudwatch:namespace": "Glue Data Quality"
        }
      }
    },
    {
      "Sid": "AllowS3PutObjectToWriteTaskResults",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject*"
      ],
      "Resource": "arn:aws:s3:::<YOUR-BUCKET-NAME>/*"
    }
  ]
}
```

Introducción a AWS Glue Data Quality para el Data Catalog

En esta sección de introducción se incluyen instrucciones para ayudarlo a empezar a utilizar AWS Glue Data Quality en la consola de AWS Glue. Aprenderá a completar tareas esenciales, como generar recomendaciones de reglas de la calidad de los datos y evaluar un conjunto de reglas en función de sus datos.

Temas

- Requisitos previos
- Ejemplo paso a paso
- Generar recomendaciones de reglas
- Recomendaciones de reglas de monitoreo
- Edición de los conjuntos de reglas recomendados
- Creación de un nuevo conjunto de reglas
- Ejecute un conjunto de reglas para evaluar la calidad de los datos
- Visualice la puntuación de la calidad de los datos y los resultados
- Temas relacionados de

Requisitos previos

Antes de usar AWS Glue Data Quality, debe estar familiarizado con el uso del Data Catalog y de los rastreadores en AWS Glue. Con AWS Glue Data Quality, puede evaluar la calidad de las tablas de una base de datos de un Data Catalog. También necesitará lo siguiente:

- Una tabla en el Data Catalog para evaluar su conjunto de reglas de la calidad de los datos.
- Un rol de IAM para AWS Glue que se proporciona al generar recomendaciones de reglas o ejecutar una tarea de calidad de datos. Este rol debe tener permiso para acceder a los recursos que requieren diversos procesos de AWS Glue Data Quality para ejecutarse en su nombre. Entre estos recursos, se incluyen AWS Glue, Amazon S3 y CloudWatch. Para ver ejemplos de políticas que incluyen los permisos mínimos para AWS Glue Data Quality, consulte Ejemplos de políticas de IAM.

  Para obtener más información sobre los roles de IAM para AWS Glue, consulte Creación de una política de IAM para el servicio de AWS Glue y Creación de un rol de IAM para el servicio AWS Glue. También puede ver una lista de todos los permisos de AWS Glue específicos para la calidad de los datos en Autorización para acciones de AWS Glue Data Quality.
- Una base de datos con al menos una tabla que contiene una variedad de datos. La tabla utilizada en este tutorial lleva el nombre `yyz-tickets`, junto con la tabla `tickets`. Estos datos son una recopilación de información de disponibilidad pública de la ciudad de Toronto sobre multas de estacionamiento. Si crea su propia tabla, asegúrese de rellenarla con una variedad de datos válidos para obtener el mejor conjunto de reglas recomendadas.

Ejemplo paso a paso

Para ver un ejemplo paso a paso con conjuntos de datos de muestra, consulte la entrada del blog de Calidad de datos de AWS Glue.

Generar recomendaciones de reglas

Las recomendaciones de reglas facilitan la introducción a la calidad de los datos sin necesidad de escribir código. Con Calidad de datos de AWS Glue, puede analizar sus datos, identificar reglas y crear un conjunto de reglas que puede evaluar en una tarea de calidad de datos. Las ejecuciones de recomendaciones se eliminan automáticamente después de 90 días.

Para generar recomendaciones sobre reglas de calidad de datos

1. Abra la consola de AWS Glue en https://console.aws.amazon.com/glue/.
2. Elija Tables (Tablas) en el panel de navegación. A continuación, elija la tabla para generar recomendaciones de reglas de la calidad de los datos.
3. En la página de detalles de la tabla, seleccione la pestaña Calidad de los datos para acceder a las reglas y configuraciones de la Calidad de datos de AWS Glue para la tabla.
4. En la pestaña Calidad de los datos, seleccione Agregar reglas y supervisar la calidad de los datos.
5. En la página Generador de conjuntos de reglas, una alerta en la parte superior de la página pedirá que inicie una tarea de recomendación si no se ejecuta ninguna recomendación de reglas.
6. Seleccione Recomendar reglas para abrir el modal e ingrese sus parámetros para la tarea de recomendación.
7. Elija un rol de IAM con acceso a AWS Glue. Este rol debe tener permiso para acceder a los recursos que requieren diversos procesos de Calidad de datos de AWS Glue para ejecutarse en su nombre.
8. Una vez completados los campos según sus preferencias, seleccione Recomendar reglas para iniciar la ejecución de la tarea de recomendación. Si las ejecuciones recomendadas están en curso o ya se completaron, puede gestionar las ejecuciones en esta alerta. Puede que tenga que actualizar la alerta para ver el cambio de estado. Las tareas de recomendación completadas y en curso aparecen en la página Historial de ejecuciones, que muestra todas las ejecuciones de recomendaciones de los últimos 90 días.

Qué significan las reglas recomendadas

Calidad de datos de AWS Glue genera reglas basadas en los datos de cada columna de la tabla de entrada. Utiliza las reglas para identificar los posibles límites por los que se pueden filtrar los datos para mantener los requisitos de calidad. La siguiente lista de reglas generadas incluye ejemplos que son útiles para entender qué significan las reglas y qué podrían hacer cuando se apliquen a los datos.

Para obtener una lista completa de los tipos de reglas del lenguaje de definición de calidad de datos (DQDL) generados, consulte la referencia de tipos de reglas del DQDL.

- `IsComplete "SET_FINE_AMOUNT"` -La regla `IsComplete` verifica que la columna esté rellenada en cualquier fila determinada. Use esta regla para etiquetar las columnas como no opcionales en los datos.
- `Uniqueness "TICKET_NUMBER" > 0.95` -La regla `Uniqueness` verifica que los datos de la columna cumplan con algún umbral de unicidad. En este ejemplo, se determinó que los datos que rellenan una fila para `"TICKET_NUMBER"` tenían, como máximo, un 95 % de contenido idéntico al de todas las demás filas, que es lo que sugiere esta regla.
- `ColumnValues "PROVINCE" in ["ON", "QC", "AB", "NY",...]` -La regla `ColumnValues` define valores válidos para la columna en función del contenido de la columna existente. En este ejemplo, los datos de cada fila son una matrícula de dos letras para un estado o una provincia.
- `ColumnLength "INFRACTION_DESCRIPTION" between 15 and 31` -La regla `ColumnLength` impone una restricción de longitud a los datos de una columna. Esta regla se genera a partir de los datos de la muestra en función de las longitudes mínima y máxima registradas para una columna de cadenas.

Recomendaciones de reglas de monitoreo

Cuando se ejecuten las recomendaciones de reglas de calidad de los datos, la página Agregar reglas y supervisar la calidad de los datos muestra información y acciones adicionales que puede realizar en la barra superior.

Cuando las recomendaciones de reglas estén en curso, puede seleccionar Detener la ejecución antes de que finalice la tarea de recomendación. Mientras la tarea esté en curso, verá el estado, en curso y la fecha y hora en que se inició la ejecución.

Cuando se hayan completado las recomendaciones de reglas, la barra de recomendaciones de reglas mostrará el número de reglas recomendadas, el estado de la última ejecución de la recomendación y la fecha y hora en que finalizó.

Para agregar reglas recomendadas, seleccione Insertar recomendación de regla. Para ver las reglas recomendadas anteriormente, selecciona una fecha específica. Para ejecutar una nueva recomendación, seleccione Más acciones y, a continuación, elija Reglas recomendadas.

Establezca la configuración predeterminada mediante la selección de Administrar la configuración de usuario. Puede establecer la ruta predeterminada de Amazon S3 para almacenar conjuntos de reglas o configurar un rol predeterminado para ejecutar el Data Catalog.

Edición de los conjuntos de reglas recomendados

Como Calidad de datos de AWS Glue genera reglas basadas en los datos existentes que tiene disponibles, es posible que vea algunas reglas inesperadas o no deseadas en las sugerencias automatizadas. Para aprovechar al máximo los conjuntos de reglas recomendados, debe evaluarlos y modificarlos. En este paso del tutorial, debe tomar las reglas generadas en el paso anterior y ajustarlas para aplicar características más restrictivas a algunos datos. También flexibiliza otras reglas para garantizar que más adelante se puedan agregar datos correctos y únicos.

Editar un conjunto de reglas sugerido

1. En la consola de AWS Glue, elija Data Catalog y, a continuación, seleccione Tablas de bases de datos en el panel de navegación. Elija la tabla `tickets`.
2. En la página de detalles de la tabla, seleccione la pestaña Calidad de los datos para acceder a las opciones de la calidad de los datos de AWS Glue para la tabla.
3. En la sección Conjuntos de reglas, seleccione el conjunto de reglas generado en Generar recomendaciones de reglas.
4. Elija Acciones y, a continuación, elija Editar en la ventana de la consola. El editor de conjuntos de reglas se carga en la consola. Incluye un panel de edición para las reglas y una referencia rápida para el DQDL.
5. Elimine la línea 2 del guion. Esto flexibiliza el requisito de limitar el tamaño de la base de datos a un número determinado de filas. Tras la edición, el archivo debe contener lo siguiente en las líneas 1 a 3:

```
Rules = [
    IsComplete "TAG_NUMBER_MASKED",
    ColumnLength "TAG_NUMBER_MASKED" between 6 and 9,
```

6. Elimine la línea 25 del guion. Esto flexibiliza el requisito de que el 96 % de las provincias registradas estén ON. Tras la edición, el archivo debe contener lo siguiente desde la línea 24 hasta el final del conjunto de reglas:

```
ColumnValues "PROVINCE" in ["ON", "QC", "AB", "NY", "AZ", "NS", "BC", "MI", "PQ",
 "MB", "PA", "FL", "SK", "NJ", "OH", "NB", "IL", "MA", "CA",
    "VA", "TX", "NF", "MD", "PE", "CT", "NC", "GA", "IN", "OR", "MN", "TN", "WI",
 "KY", "MO", "WA", "NH", "SC", "CO", "OK", "VT", "RI", "ME", "AL",
    "YT", "IA", "DE", "AR", "LA", "XX", "WV", "MT", "KS", "NT", "DC", "NV", "NE",
 "UT", "MS", "NM", "ID", "SD", "ND", "AK", "NU", "GO", "WY", "HI"],
```

```
ColumnLength "PROVINCE" = 2
]
```

7. Cambie la línea 14 por lo siguiente:

```
IsComplete "TIME_OF_INFRACTION",
```

   Esto refuerza el requisito de la columna al limitar la base de datos únicamente a los tickets que contengan una fecha de infracción registrada. En este conjunto de datos, siempre debe considerar que los tickets sin una hora de infracción registrada son datos no válidos. Esto es diferente a las situaciones en las que la partición o la transformación podrían ser más adecuadas para seguir utilizando los datos o inspeccionándolos a fin de determinar una regla de calidad.

8. Seleccione Actualizar conjunto de reglas en la parte inferior de la página de la consola.

Creación de un nuevo conjunto de reglas

Un conjunto de reglas es un grupo de reglas de la calidad de los datos que se evalúa en relación con los datos. En la consola de AWS Glue, puede crear conjuntos de reglas personalizados mediante el lenguaje de definición de calidad de los datos (DQDL).

Para crear un conjunto de reglas de calidad de datos

1. En la consola de AWS Glue, elija Data Catalog, luego Bases de datos y, a continuación, Tablas en el panel de navegación. Seleccione la tabla `tickets`.
2. Abra la pestaña Data quality (Calidad de datos).
3. En la sección Conjuntos de reglas, elija Crear regla. El editor DQDL se inicia en la consola. Cuenta con un área de texto para la edición directa y una referencia rápida para las reglas del DQDL y el esquema de la tabla.
4. Comience a agregar reglas al área de texto del editor DQDL. Puede escribir reglas directamente desde este tutorial o utilizar la característica generador de reglas DQDL del editor de reglas de calidad de datos.

> **Note**
>
> Cómo usar el generador de reglas DQDL
>
> 1. Seleccione un tipo de regla de la lista y elija el signo más para insertar un ejemplo de sintaxis en el panel del editor.
> 2. Intercambie los nombres de las columnas del marcador de posición por sus propios nombres de columna. Los nombres de las columnas de la tabla están disponibles en la pestaña Esquema.
> 3. Actualice el parámetro de expresión como crea conveniente. Para obtener una lista completa de las expresiones que admite DQDL, consulte Expressions.

Por ejemplo, las siguientes reglas son restricciones para la validación de datos de la columna `ticket_number` de la tabla `tickets`. Para agregar las siguientes reglas, utilice el generador de reglas DQDL o edite directamente el conjunto de reglas:

```
IsComplete "ticket_number",
IsUnique "ticket_number",
ColumnValues "ticket_number" > 9000000000
```

5. Proporcione un nombre para su nuevo conjunto de reglas en el campo Nombre del conjunto de reglas.
6. Seleccione Guardar conjunto de reglas.

Evaluación de la calidad de los datos en varios conjuntos de datos

Puede configurar reglas de calidad de datos en varios conjuntos de datos mediante los conjuntos de reglas ReferentialIntegrity y DatasetMatch. ReferentialIntegrity comprueba si los datos del conjunto de datos principal están presentes en otros conjuntos de datos.

Para agregar un conjunto de datos de referencia, seleccione la pestaña Esquema y, a continuación, seleccione Actualizar tablas de referencia. Se pedirá que seleccione una base de datos y una tabla. Puede agregar la tabla y, a continuación, configurar las reglas de calidad de los datos. Los tipos de reglas como AggregateMatch, RowCountMatch, ReferentialIntegrity, SchemaMatch y DatasetMatch permiten realizar comprobaciones de calidad de datos en varios conjuntos de datos.

Ejecute un conjunto de reglas para evaluar la calidad de los datos

Al ejecutar una tarea de calidad de datos, Calidad de datos de AWS Glue evalúa un conjunto de reglas mediante la comparación con los datos y calcula una puntuación de la calidad de los datos. Esta puntuación representa el porcentaje de reglas de la calidad de los datos que se aprobaron para la entrada.

Para ejecutar una tarea de calidad de datos

1. En la consola de AWS Glue, elija Data Catalog, luego Bases de datos y, a continuación, Tablas en el panel de navegación. Seleccione la tabla `tickets`.
2. Seleccione la pestaña Calidad de los datos.
3. En la lista Conjuntos de reglas, elija el conjunto de reglas que quiere evaluar con respecto a la tabla. Para este paso, recomendamos que utilice un conjunto de reglas que ya haya escrito o modificado en lugar de generar reglas. Elija Ejecutar.
4. En el modal, elija su rol de IAM. Este rol debe tener permiso para acceder a los recursos que requieren diversos procesos de Calidad de datos de AWS Glue para ejecutarse en su nombre. Puede guardar el rol de IAM como predeterminado o modificarlo desde la página de Ajuste predeterminado.
5. En Data quality actions (Acciones de calidad de datos), elija si quiere publicar las métricas en Amazon CloudWatch. Cuando se selecciona esta opción, Calidad de datos de AWS Glue publica métricas que indican la cantidad de reglas que se aprobaron y la cantidad de reglas con errores. Para tomar medidas con respecto a las métricas almacenadas de esta forma, puede utilizar las alarmas de CloudWatch. También se publican las métricas clave en Amazon EventBridge para que pueda configurar las alertas. Para obtener más información, consulte Configurar alertas, implementaciones y programación.
6. En Frecuencia de ejecución, seleccione Ejecutar bajo demanda o programa el conjunto de reglas. Al programar un conjunto de reglas, se solicitará el nombre de la tarea. La programación se creará en Amazon EventBridge. Puede editar su horario en Amazon EventBridge.
7. Para guardar los resultados de calidad de datos en Amazon S3, elija una Ubicación de resultados de calidad de datos. El rol de IAM que seleccionó anteriormente para esta tarea debe tener acceso de escritura a esta ubicación.
8. En Configuraciones adicionales, ingrese la cantidad solicitada de trabajadores que quiere que AWS Glue asigne a su tarea de calidad de datos.
9. Si lo desea, puede configurar un filtro en el origen de datos. Esto lo ayuda a reducir los datos que lee. También puede usar un filtro para ejecutar validaciones incrementales al seleccionar la

información de la partición y pasarla como parámetros mediante llamadas a la API. Para mejorar el rendimiento, puede proporcionar un predicado de partición.

10. Elija Ejecutar. Debería ver la nueva tarea en la lista Data quality task runs (Ejecuciones de tareas de calidad de datos). Cuando la columna de estado de ejecución de la tarea aparezca como Completada, podrá ver los resultados de la puntuación de calidad. Puede que tenga que actualizar la ventana de la consola para que el estado se actualice correctamente.
11. Para ver la columna con los detalles de los resultados de calidad de los datos, seleccione el icono “+” para expandir el conjunto de reglas. Los resultados muestran las reglas que se aprobaron y las que no se aprobaron en la evaluación, además del motivo del fallo de la regla.

Visualice la puntuación de la calidad de los datos y los resultados

Para ver la última ejecución de todos los conjuntos de reglas creados

1. En la consola de AWS Glue, seleccione Tables (Tablas) en el panel de navegación. A continuación, elija la tabla para ejecutar una tarea de calidad de datos.
2. Seleccione la pestaña Calidad de los datos.
3. La instantánea de la calidad de los datos muestra una tendencia general de las ejecuciones a lo largo del tiempo. De forma predeterminada, se muestran las últimas 10 ejecuciones de todos los conjuntos de reglas. Para filtrar por conjunto de reglas, seleccione el que desee en la lista desplegable. Si hay menos de 10 ejecuciones, se muestran todas las ejecuciones completadas disponibles.
4. En la tabla de calidad de los datos, se muestra cada conjunto de reglas con su última ejecución (si la hay), junto con la puntuación. Al expandir el conjunto de reglas, se muestran las reglas que están en ese conjunto de reglas, junto con los resultados de las reglas de esa ejecución.

Para ver la última ejecución de un conjunto de reglas concreto

1. En la consola de AWS Glue, seleccione Tables (Tablas) en el panel de navegación. A continuación, elija la tabla para ejecutar una tarea de calidad de datos.
2. Seleccione la pestaña Calidad de los datos.
3. En la tabla calidad de los datos, elija un conjunto de reglas específico.
4. En la página detalles del conjunto de reglas, seleccione la pestaña Historial de ejecuciones.

Todas las ejecuciones de evaluación de este conjunto de reglas en particular se muestran en la tabla de esta pestaña. Puede ver el historial de las puntuaciones y el estado de las ejecuciones.

5. Para ver más información sobre una ejecución concreta, elija el ID de ejecución para ir a la página de detalles de la ejecución de evaluación. En esta página, puede ver información específica sobre la ejecución y más detalles sobre el estado de los resultados de las reglas individuales.

Temas relacionados de

- Referencia de tipo de regla de DQDL
- Referencia del lenguaje de definición de calidad de datos (DQDL)

Evaluación de la calidad de los datos con AWS Glue Studio

Calidad de datos de AWS Glue evalúa y supervisa la calidad de sus datos en función de las reglas que defina. Esto facilita la identificación de los datos que requieren acción. En AWS Glue Studio, puede agregar nodos de calidad de datos a su trabajo visual para crear reglas de calidad de datos en las tablas del catálogo de datos. Puede monitorear y evaluar los cambios aplicados en sus conjuntos de datos a medida que evolucionan. Para obtener información general sobre cómo trabajar con Data Quality de AWS Glue en AWS Glue Studio, consulte el siguiente vídeo.

A continuación se indican los pasos generales para trabajar con Calidad de datos de AWS Glue:

1. Cree reglas de calidad de datos: cree un conjunto de reglas de calidad de datos con el generador de DQDL; para ello, seleccione los conjuntos de reglas integrados que configure.
2. Configure un trabajo de calidad de datos: defina acciones en función de los resultados de calidad de los datos y las opciones de salida.
3. Guarde y ejecute un trabajo de calidad de datos: cree y ejecute un trabajo. Al guardar el trabajo, se guardarán los conjuntos de reglas que creó para el trabajo.
4. Supervise y revise los resultados de calidad de los datos: revise los resultados de calidad de los datos una vez finalizada la ejecución del trabajo. Si lo desea, programe el trabajo para una fecha futura.

Ventajas

Los analistas de datos, ingenieros de datos y científicos de datos pueden utilizar el nodo de evaluación de la calidad de los datos en AWS Glue Studio para analizar, configurar, supervisar y mejorar la calidad de los datos desde el editor de trabajos visuales. Las ventajas de utilizar el nodo de calidad de datos incluyen lo siguiente:

- Puede detectar problemas de calidad de los datos: puede crear reglas que comprueben las características de los conjuntos de datos para determinar si hay problemas.
- Empezar a trabajar es fácil: puede empezar con reglas y acciones prediseñadas.
- Integración estrecha: puede utilizar nodos de calidad de datos en AWS Glue Studio porque Calidad de datos de AWS Glue se ejecuta sobre el Catálogo de datos de AWS Glue.

Evaluación de la calidad de los datos para los trabajos de ETL en AWS Glue Studio

En este tutorial, empezará a utilizar la calidad AWS Glue de los datos en AWS Glue Studio. Aprenderá a realizar lo siguiente:

- El conjunto de reglas se crea mediante lenguaje de definición de calidad de datos (DQDL).
- Especificar las acciones de calidad de datos, los datos que se van a generar y la ubicación de la salida de los resultados de calidad de datos.
- Revisar los resultados de calidad de datos.

Para practicar con un ejemplo, consulte la entrada del blog Introducción a la calidad de los datos AWS para tuberías ETL.

Paso 1: agregar el nodo de transformación de evaluación de Calidad de datos al trabajo visual

En este paso, agrega el nodo de evaluación de Calidad de datos al editor de trabajos visuales.

Para agregar el nodo de calidad de datos

1. En la consola AWS Glue Studio, seleccione Visual con un origen y un destino en la sección Crear trabajo y, luego, seleccione Crear.

2. Elija un nodo al que desee aplicar la transformación de calidad de los datos. Normalmente será un nodo de transformación o un origen de datos.
3. Abra el panel de recursos de la izquierda mediante la selección del icono “+”. También puede escribir calidad de los datos en la barra de búsqueda y, luego, elegir Evaluar la calidad de los datos en los resultados de la búsqueda.
4. El editor de trabajos visual mostrará el nodo de transformación Evaluar la calidad de los datos ramificándose a partir del nodo que seleccionó. En la parte derecha de la consola, la pestaña Transform (Transformación) se abre automáticamente. Si necesita cambiar el nodo principal, seleccione la pestaña Propiedades del nodo y, luego, elija el nodo principal en el menú desplegable.

   Al elegir un nuevo nodo principal, se establece una nueva conexión entre el nodo principal y el nodo Evaluate Data Quality (Evaluar la calidad de los datos). Elimine los nodos principales no deseados. Solo se puede conectar un nodo principal a un nodo Evaluate Data Quality (Evaluar la calidad de los datos).
5. La transformación de evaluación de la calidad de los datos admite varias fuentes, por lo que puede validar las reglas de calidad de los datos en varios conjuntos de datos. Entre las reglas que admiten varios conjuntos de datos se incluyen ReferentialIntegrity, DatasetMatch, SchemaMatch, RowCountMatch y AggregateMatch.

   Al agregar varias entradas a la transformación de evaluación de la calidad de los datos, debe seleccionar la entrada “principal”. La entrada principal es el conjunto de datos para el que desea validar la calidad de los datos. Todos los demás nodos o entradas se tratan como referencias.

   Puede utilizar la transformación de evaluación de la calidad de los datos para identificar registros específicos que no superaron las comprobaciones de calidad de los datos. Recomendamos que elija el conjunto de datos principal, ya que las nuevas columnas que indican los registros incorrectos se agregan al conjunto de datos principal.
6. Puede especificar alias para los orígenes de datos de entrada. Los alias proporcionan otra forma de hacer referencia al origen de entrada cuando se utiliza la regla de ReferentialIntegrity. Ya que solo se puede designar un origen de datos como fuente principal, cada origen de datos adicional que agregue necesitará un alias.

   En el siguiente ejemplo, la regla ReferentialIntegrity especifica el origen de datos de entrada por el nombre del alias y realiza una comparación uno a uno con el origen de datos principal.

```
Rules = [
```

```
 ReferentialIntegrity “Aliasname.name” = 1
]
```

Paso 2: crear una regla con DQDL

En este paso, va a crear una regla con DQDL. Para este tutorial, creará una sola regla con el tipo de regla Integridad. Este tipo de regla comprueba el porcentaje de valores completos (no nulos) de una columna con respecto a una expresión determinada. Para obtener más información sobre cómo usar DQDL, consulte DQDL.

1. En la pestaña Transformación, haga clic en el botón de inserción para agregar un tipo de regla. De esta forma, se agrega el tipo de regla al editor de reglas, donde puede ingresar los parámetros de la regla.

   > **Note**
   >
   > Al editar las reglas, asegúrese de que estén entre corchetes y separadas por comas. Por ejemplo, una expresión de regla completa tendrá el siguiente aspecto:
   >
   > ```
   > Rules= [
   >     Completeness "year">0.8, Completeness "month">0.8
   > ]
   > ```
   >
   > En este ejemplo se especifica el parámetro de integridad de las columnas denominadas “año” y “mes”. Estas columnas deben estar “completas” en más del 80 % o tener datos en más del 80 % de las instancias de cada columna correspondiente para que la regla se apruebe.

   En este ejemplo, busque e inserte el tipo de regla Completeness (Integridad). De esta forma, se agregará el tipo de regla al editor de reglas. Este tipo de regla tiene la siguiente sintaxis: `Completeness <COL_NAME> <EXPRESSION>`.

   La mayoría de los tipos de reglas requieren que proporcione una expresión como parámetro para crear una respuesta booleana. Para obtener más información sobre las expresiones de

DQDL admitidas, consulte Expresiones de DQDL. A continuación, agregará el nombre de la columna.

2. En el generador de reglas de DQDL, haga clic en la pestaña Esquema. Utilice la barra de búsqueda para localizar el nombre de la columna en el esquema de entrada. El esquema de entrada muestra el nombre de la columna y el tipo de datos.
3. En el editor de reglas, haga clic a la derecha del tipo de regla para insertar el cursor en el lugar en el que se insertará la columna. También puede escribir el nombre de la columna en la regla.

   Por ejemplo, en la lista de columnas de la lista del esquema de entrada, haga clic en el botón de inserción situado junto a la columna (en este ejemplo, year). De esta forma, se agregará la columna a la regla.
4. A continuación, en el editor de reglas, agregue una expresión para evaluar la regla. Dado que el tipo de regla Integridad compara el porcentaje de valores completos (no nulos) de una columna con una expresión determinada, ingrese una expresión como `> 0.8`. Esta regla comprobará la columna si tiene más del 80 % de valores completos (no nulos).

Paso 3: configurar las acciones y la salida de Calidad de datos

Tras crear las reglas de calidad de los datos, puede seleccionar opciones adicionales para especificar la salida del nodo de calidad de datos.

1. En Data quality transform output (Salida de la transformación de calidad de datos), elija una de las siguientes opciones:

   - Datos originales: elija generar los datos de entrada originales. Al elegir esta opción, se agrega un nuevo nodo secundario, “rowLevelOutcomes”, al trabajo. El esquema coincide con el esquema del conjunto de datos principal que se pasó como entrada a la transformación. Esta opción es útil si solo quieres pasar los datos y fallar en el trabajo cuando se producen problemas de calidad.

     Otro caso de uso es cuando se desean detectar registros defectuosos que no superaron las comprobaciones de calidad de los datos. Para detectar registros incorrectos, elija la opción Agregar nuevas columnas para indicar errores en la calidad de los datos. Esta acción agrega cuatro columnas nuevas al esquema de la transformación “rowLevelOutcomes”.

     - DataQualityRulesPass (Matriz de cadenas): proporciona una serie de reglas que han superado las comprobaciones de calidad de los datos.

- DataQualityRulesFail (Matriz de cadenas): proporciona una serie de reglas que no superaron las comprobaciones de calidad de los datos.
- DataQualityRulesSkip (Matriz de cadenas): proporciona una serie de reglas que se omitieron. Las siguientes reglas no pueden identificar los registros de errores, ya que se aplican a nivel de conjunto de datos.
  - AggregateMatch
  - ColumnCount
  - ColumnExists
  - ColumnNamesMatchPattern
  - CustomSql
  - RowCount
  - RowCountMatch
  - StandardDeviation
  - Media
  - ColumnCorrelation
- DataQualityEvaluationResult: proporciona el estado “Aprobado” o “Fallido” a nivel de fila. Tenga en cuenta que los resultados generales pueden ser FALLA, pero es posible que se apruebe un registro determinado. Por ejemplo, es posible que la regla RowCount haya fallado, pero es posible que todas las demás reglas se hayan aplicado correctamente. En esos casos, el estado de este campo es “Aprobado”.

2. Resultados de calidad de datos: elija generar las reglas configuradas y el estado correspondiente que indique que han aprobado o suspendido. Esta opción es útil para escribir los resultados en Amazon S3 u otras bases de datos.
3. Configuración de la salida de calidad de datos (Opcional): elija la Configuración de la salida de calidad de datos para mostrar el campo Ubicación del resultado de calidad de datos. Luego, haga clic en Explorar para buscar una ubicación de Amazon S3 y establecerla como destino de la salida de la calidad de los datos.

Paso 4. Configurar acciones de calidad de datos

Las acciones permiten publicar métricas en CloudWatch o detener trabajos en función de criterios específicos. Las acciones solo están disponibles después de crear una regla. Al elegir esta opción,

también se publican las mismas métricas en Amazon EventBridge. Puedes usar estas opciones para crear alertas para las notificaciones.

- En caso de error en un conjunto de reglas: puede elegir qué hacer si un conjunto de reglas falla mientras se ejecuta el trabajo. Si desea que el trabajo no se apruebe si la calidad de los datos no es buena, elija cuándo no debe aprobarse el trabajo mediante la selección de una de las siguientes opciones. De forma predeterminada, esta acción no está seleccionada y el trabajo finalizará su ejecución incluso si las reglas de calidad de los datos no aprueban.
  - Ninguno: si selecciona Ninguno (opción predeterminada), el trabajo no se aprobará y seguirá ejecutándose a pesar de los errores del conjunto de reglas.
  - Error en el trabajo después de cargar los datos en el destino: se produce un error en el trabajo y no se guarda ningún dato. Para guardar los resultados, elija una ubicación de Amazon S3 en la que se guarden los resultados de calidad de los datos.
  - Trabajo erróneo sin cargar los datos de destino: esta opción anula el trabajo inmediatamente cuando se produce un error en la calidad de los datos. No carga ningún destino de datos, incluidos los resultados de la transformación de la calidad de los datos.

Paso 5: ver resultados de calidad de datos

Después de ejecutar el trabajo, elija la pestaña Calidad de los datos para ver los resultados de calidad de los datos.

1. Para cada trabajo ejecutado, consulte los resultados de Calidad de datos. Cada nodo muestra un estado de calidad de datos y detalles del estado. Haga clic en un nodo para ver todas las reglas y el estado de cada una de ellas.
2. Haga clic en Descargar resultados para descargar un archivo CSV que contenga información sobre la ejecución del trabajo y los resultados de calidad de datos.
3. Si ha ejecutado más de un trabajo con resultados de calidad de los datos, puede filtrar los resultados por intervalo de fechas y horas. Haga clic en Filtrar por intervalo de fechas y horas para expandir la ventana de filtros.
4. Elija un intervalo relativo o un intervalo absoluto. En el caso de los intervalos absolutos, utilice el calendario para seleccionar una fecha e ingrese valores para la hora de inicio y la hora de finalización. Cuando haya finalizado, elija Aplicar cambios.

Generador de reglas de Calidad de datos

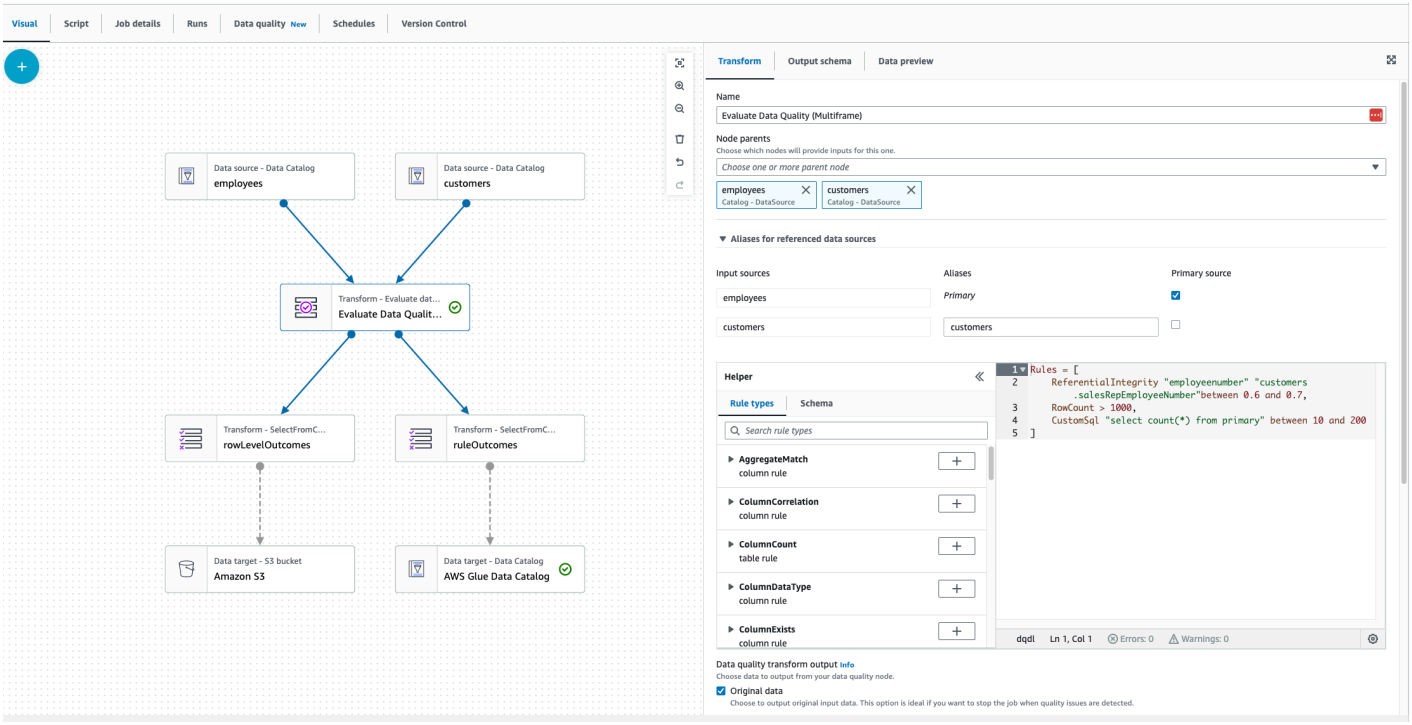
Con el generador de reglas del lenguaje de definición de calidad de datos (DQDL), puede crear reglas de calidad de datos para evaluar sus datos. Para empezar, seleccione un tipo de regla y, a continuación, especifique los parámetros en el editor de reglas. El editor de reglas también muestra cualquier error y advertencia a medida que cree las reglas.

En la Guía de DQDL, se proporciona documentación completa sobre cómo crear reglas mediante la sintaxis de DQDL, tipos de reglas integrados y ejemplos.

Nodo de evaluación de Calidad de datos

Cuando trabaje con el nodo de transformación Evaluar la calidad de los datos y el generador de reglas de DQDL, puede expandir el espacio de trabajo.

- A fin de expandir la pestaña Transformación para que ocupe toda la pantalla, seleccione el icono de expansión situado en la esquina superior derecha del panel de detalles del nodo.
- Para expandir el editor de reglas de DQDL, seleccione el icono << para expandir el editor de reglas y contraer las pestañas Tipos de reglas y Esquema.



Componentes

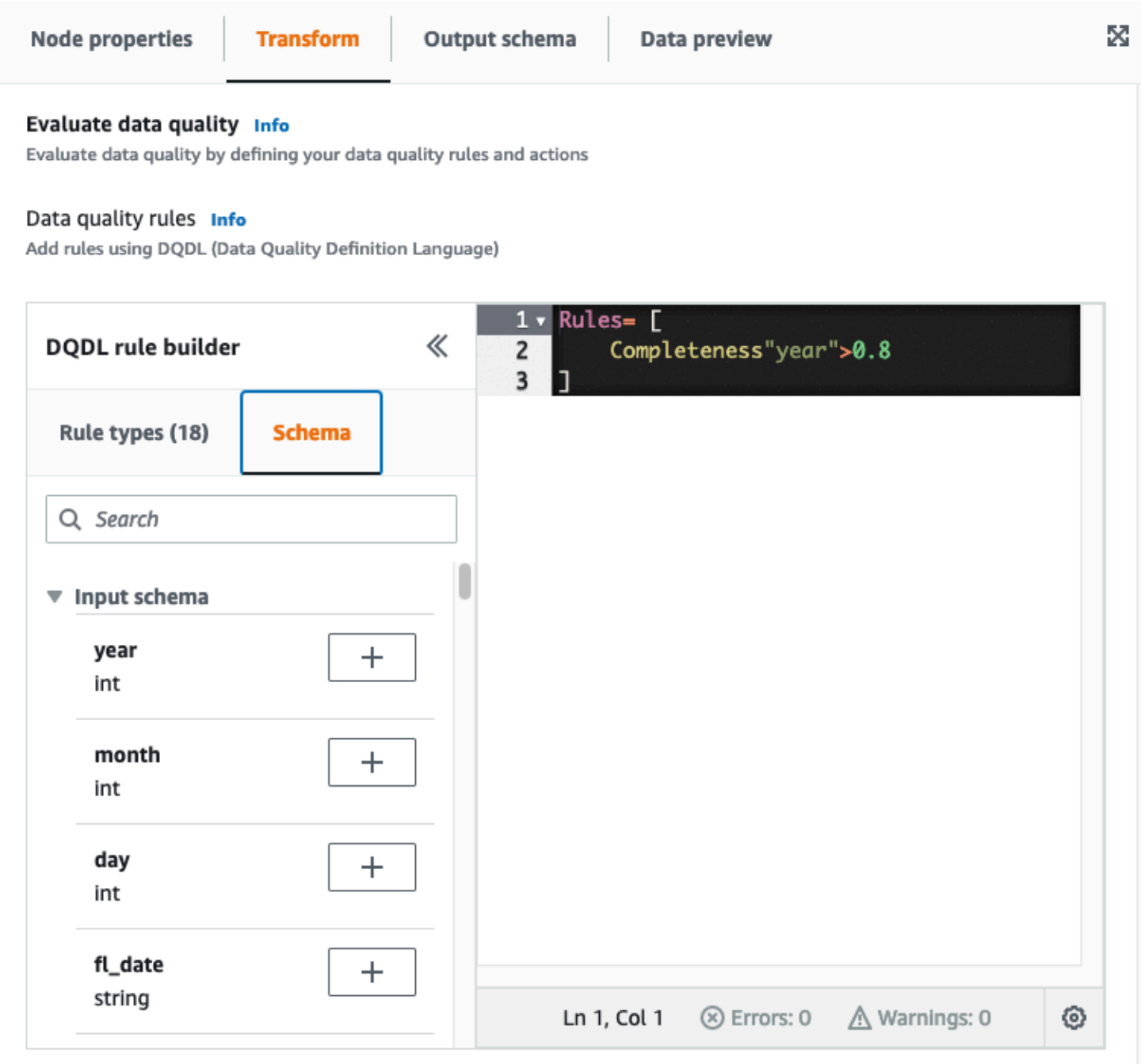
Hay 26 tipos de reglas integrados en AWS Glue Studio. Cada tipo de regla tiene una descripción y ejemplos de cómo se puede utilizar.

Tipos de reglas de calidad de datos

AWS Glue Studio proporciona tipos de reglas integrados para facilitar la creación de una regla. Para obtener más información sobre los tipos de reglas, consulte la referencia sobre los tipos de reglas de DQDL.

Esquema

En la pestaña Schema (Esquema), se muestran los nombres de las columnas y el tipo de datos del nodo principal. Se muestran los esquemas de varios nodos. Puede ver el esquema de entrada, buscar por nombre de columna e insertar la columna en el editor de reglas.



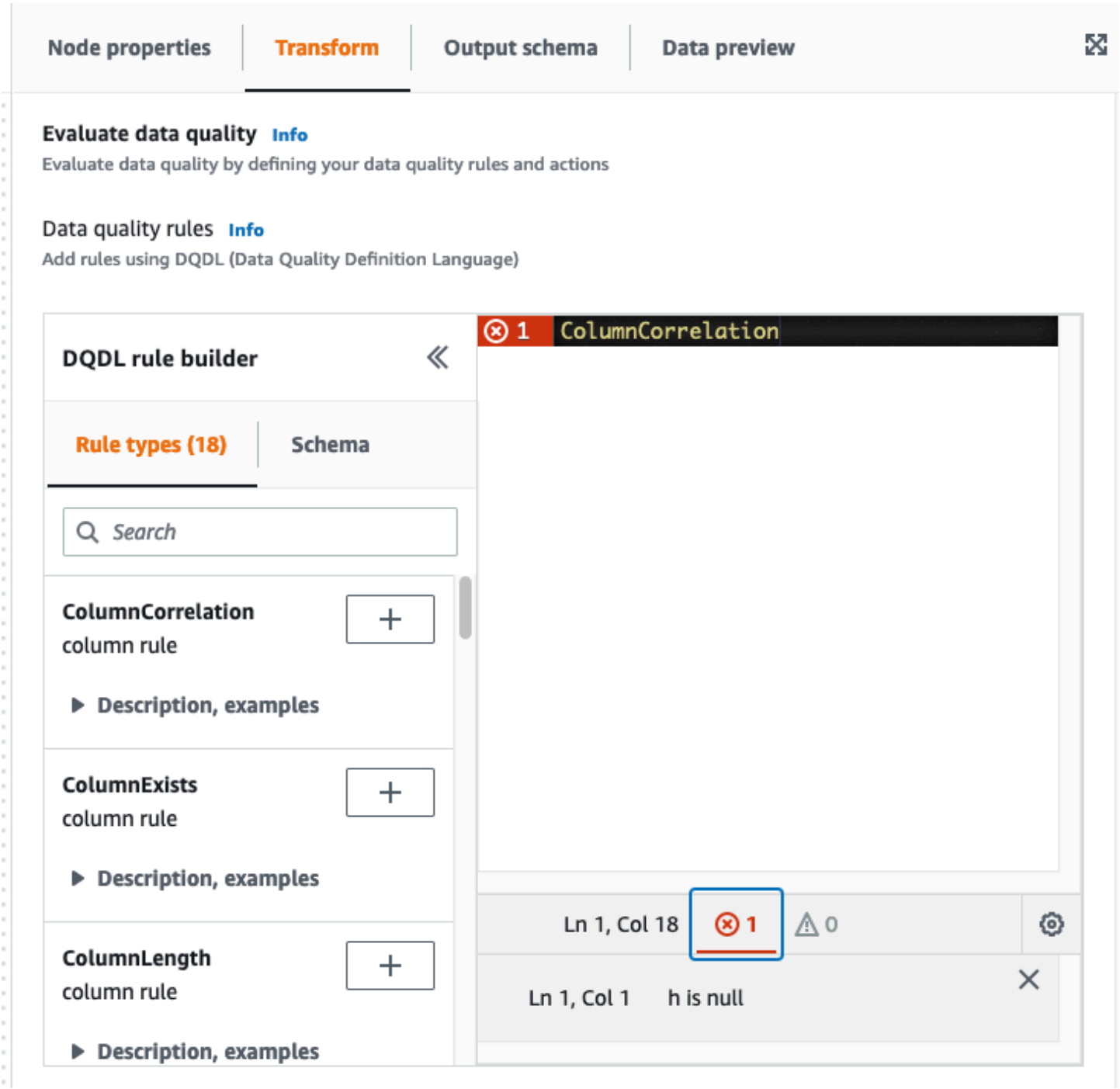
Editor de reglas

El editor de reglas es un editor de texto en el que puede escribir y editar reglas. Si selecciona un tipo de regla en el generador de reglas de DQDL, el tipo de regla se agrega al editor de reglas. A continuación, puede especificar parámetros, agregar y editar reglas según sea necesario, para lo que debe modificar el texto. AWS Glue Studio valida las reglas en el editor de reglas y muestra los errores y advertencias en caso de que haya alguna.

Errores y advertencias

Si una regla no sigue la sintaxis de las reglas de DQDL, el editor de reglas muestra varios indicadores visuales para señalar que hay un error:

- El editor de reglas muestra un icono de error en rojo en la línea que tiene el error.
- El editor de reglas muestra el número de errores junto al icono de error rojo.
- Al seleccionar la línea que tiene el error, se muestran descripciones del error y la ubicación (línea y columna) en la parte inferior del editor de reglas.



Acciones de calidad de datos

De forma predeterminada, esta acción no está seleccionada y el trabajo finalizará su ejecución incluso si las reglas de calidad de los datos no aprueban.

Elija entre las siguientes acciones. Puede utilizar acciones para publicar resultados en CloudWatch o detener trabajos en función de criterios específicos. Las acciones solo están disponibles después de crear una regla.

- Publicar los resultados en CloudWatch: cuando ejecute un trabajo, agregue los resultados a CloudWatch.
- Producir un error en el trabajo cuando se produzca un error en la calidad de los datos: si se produce un error en las reglas de calidad de los datos, también se producirá un error en el trabajo como resultado.

Salida de la transformación de calidad de datos

- Datos originales: elija generar los datos de entrada originales. Esta opción es ideal si desea detener el trabajo cuando se detecten problemas de calidad.
- Métricas de calidad de datos: elija generar las reglas configuradas y el estado correspondiente que indique que han aprobado o suspendido. Esta opción es útil si desea llevar a cabo una acción personalizada.

Configuración de la salida de calidad de datos

Para establecer la ubicación de los resultados de calidad de los datos, especifica la ubicación de Amazon S3 como destino de la salida de calidad de los datos.

Configuración de la detección de anomalías y generación de información

La calidad de los datos (DQ) de AWS Glue evalúa los datos en función de las normas de calidad de los datos que usted establece, y da información y observaciones sobre los datos a lo largo del tiempo para que pueda tomar medidas de inmediato. Dado que DQ escanea sus datos, calcula métricas estadísticas, como el recuento de filas, el máximo o el mínimo, y luego las compara con las expresiones de umbral.

Algunas de las ventajas de la detección de anomalías en la calidad de los datos incluyen:

- escaneo continuo y automatizado de datos
- detección de anomalías que puedan ser indicios de un hecho imprevisto o de una anomalía estadística
- ofrecer recomendaciones de reglas para tomar medidas en relación con las observaciones detectadas mediante la detección de anomalías en la calidad de los datos

Esto resulta útil si:

- quiere detectar anomalías en sus datos automáticamente, sin necesidad de escribir datos de calidad
- quiere perfilar sus datos y ver representaciones visuales del aspecto de los datos
- quiere hacer un seguimiento de cómo cambian sus datos a lo largo del tiempo

¿Qué observaciones puedo ver sobre mis datos?

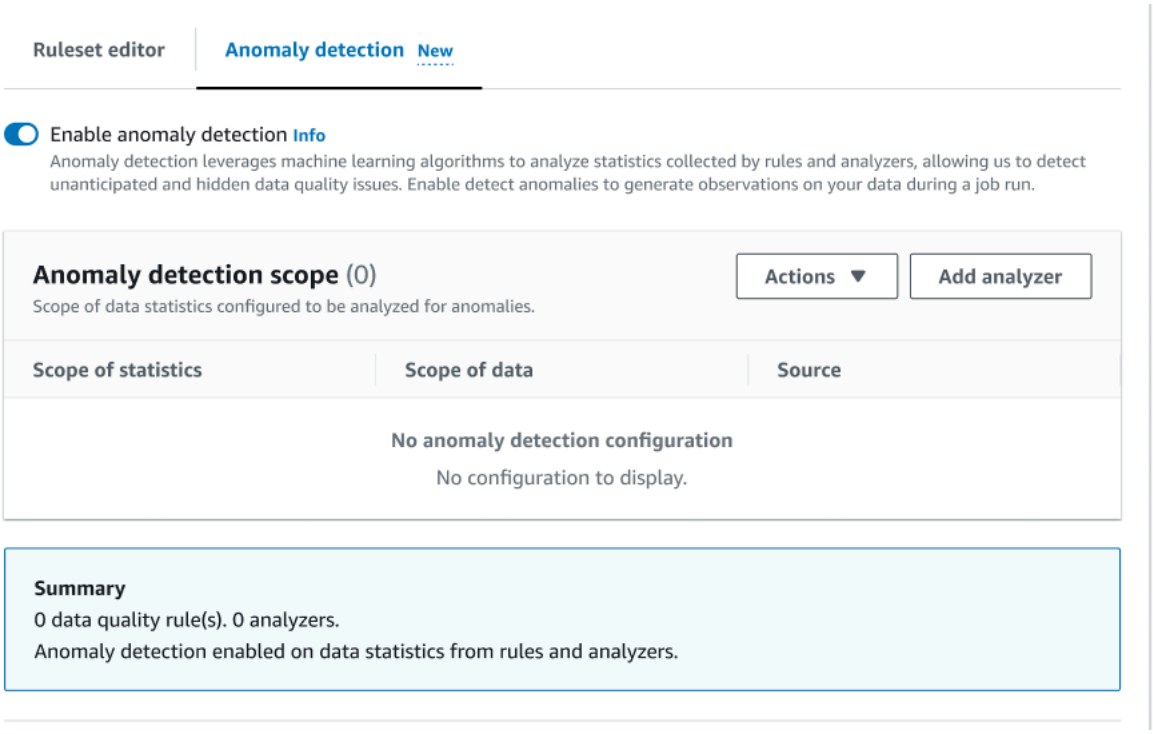
El DQ identifica los valores atípicos en las estadísticas de datos recopiladas, los cambios en los formatos de los datos, las desviaciones de los datos y los cambios de esquema. Partiendo de las observaciones, el DQ recomienda reglas de calidad de los datos que los usuarios puedan poner en práctica fácilmente. Las estadísticas incluyen integridad, unicidad, media, suma StandardDeviation, entropía y. DistinctValuesCount UniqueValueRatio

Cómo habilitar la detección de anomalías en AWS Glue Studio

Para activar la detección de anomalías, puede abrir un trabajo de AWS Glue Studio y activar la opción “Activar la detección de anomalías”. Si lo activa, podrá detectar anomalías en sus datos, ya que los analizará a lo largo del tiempo y dará estadísticas sobre sus datos y observaciones para que pueda actuar en consecuencia.

Para habilitar la detección de anomalías en AWS Glue Studio:

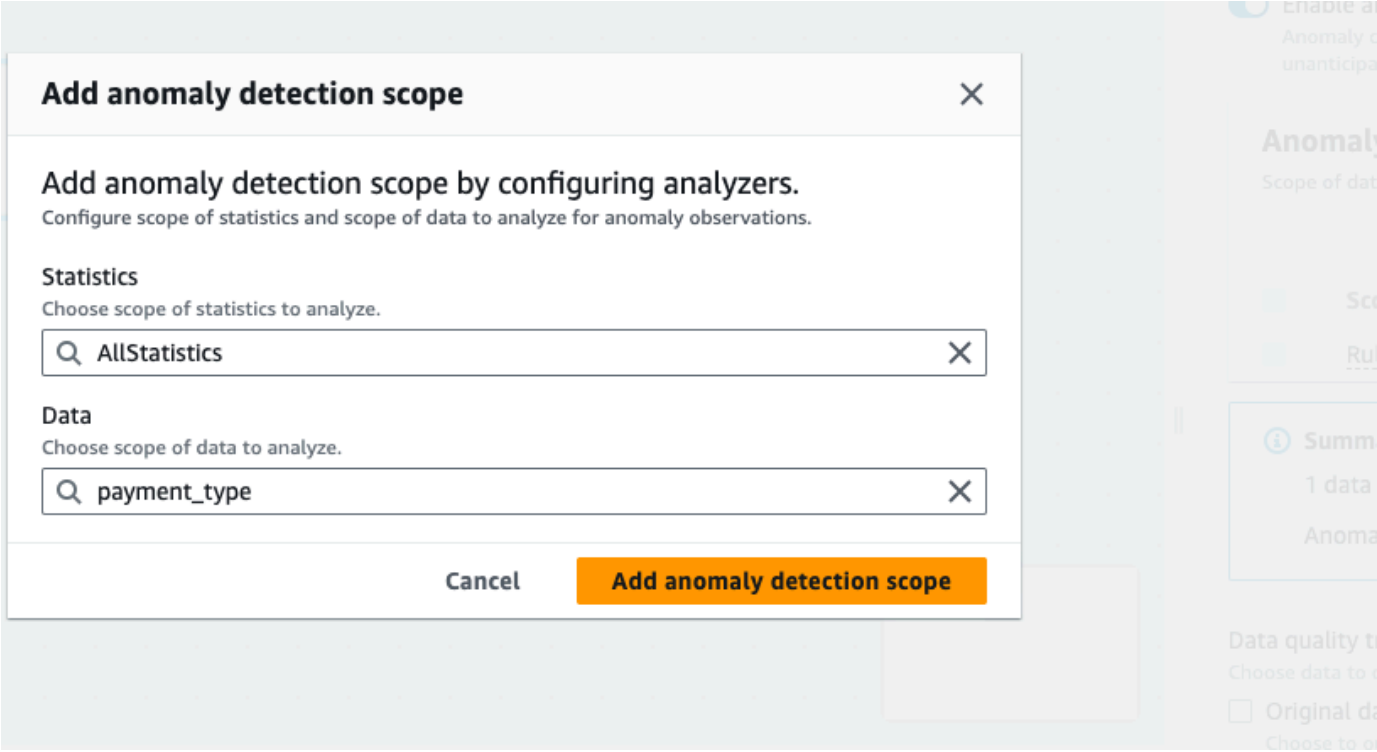
1. Elija el nodo de calidad de datos de su trabajo y, a continuación, elija la pestaña Detección de anomalías. Active “Activar la detección de anomalías”.



2. Defina los datos que se van a monitorizar en busca de anomalías seleccionando Añadir analizador. Hay dos campos que puede rellenar: Estadísticas y Datos.

   Las estadísticas son información sobre la forma de los datos y otras propiedades. Puede elegir una o varias estadísticas a la vez o elegir todas las estadísticas. Las estadísticas incluyen: integridad, unicidad, media, suma StandardDeviation, entropía y. DistinctValuesCount UniqueValueRatio

   Los datos son las columnas del conjunto de datos. Puede elegir todas las columnas o columnas individuales.



3. Elija Agregar alcance de detección de anomalías para guardar los cambios. Cuando haya creado los analizadores, podrá verlos en la sección Alcance de la detección de anomalías.

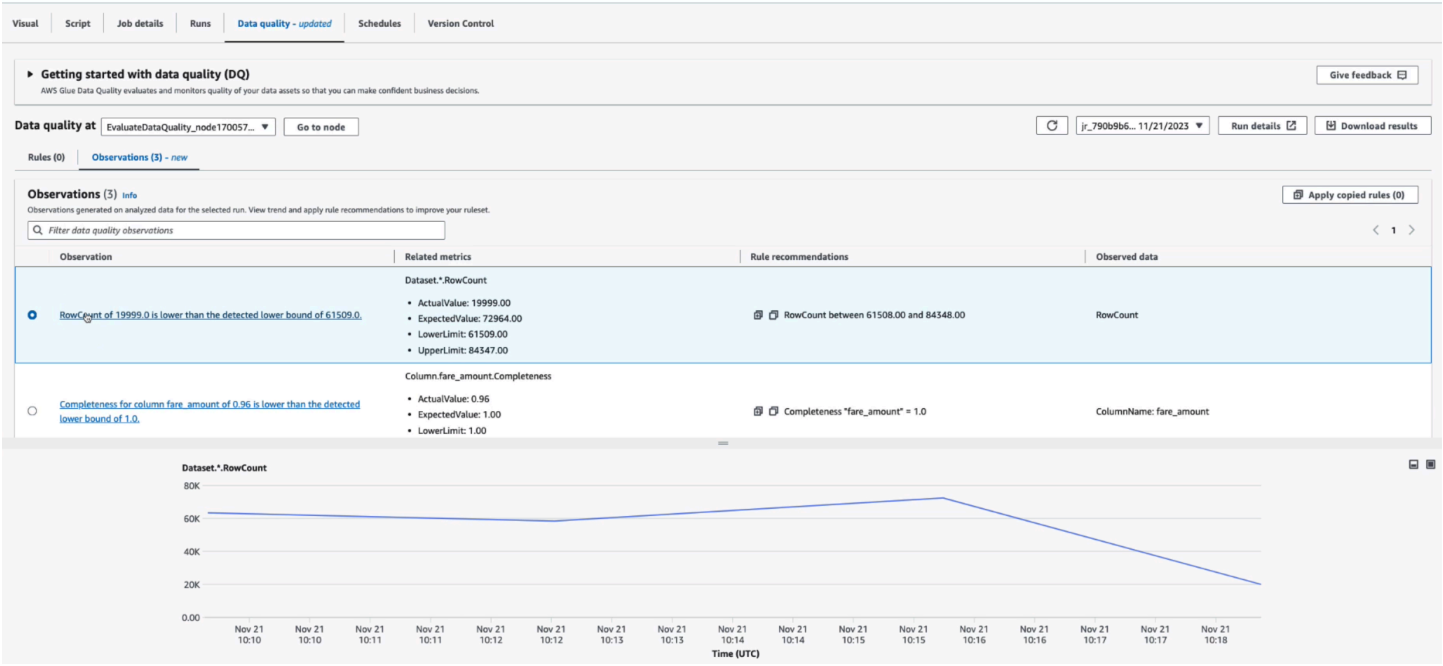
   También puede utilizar el menú Acciones para editar los analizadores, o elegir la pestaña del Editor de conjuntos de reglas y editar el analizador directamente en el bloc de notas del editor de conjuntos de reglas. Verá los analizadores que ha guardado justo debajo de las reglas que creó.

```
Rules = [
]

Analyzers = [
    Completeness “id”
]
```

   Con el conjunto de reglas actualizado y los analizadores, Calidad de los datos monitorea continuamente los datos entrantes y detecta las anomalías mediante alertas o interrupciones del trabajo en función de su configuración.

> **Note**
>
> Las observaciones se generan cuando se observan un mínimo de tres valores por estadística de datos en el conjunto de datos. Si no hay observaciones visibles, Calidad de los datos no tiene datos suficientes para generar una observación. Tras varios trabajos, Calidad de los datos puede presentar información sobre los datos y los mostrará en la sección Observaciones.

Los analizadores generan observaciones mediante la detección de anomalías en los datos y ofrecen recomendaciones para crear reglas de forma progresiva. Puede ver las observaciones si elige la pestaña Calidad de los datos. Las observaciones son específicas de cada ejecución de trabajo. Puede ver el nodo de calidad de datos específico y la ejecución del trabajo en la parte superior de la sección Observaciones. Elija un nuevo nodo o ejecución de tareas para ver las observaciones específicas de ese nodo y trabajo.



Observación: cada información se basa en una ejecución de trabajo específica configurada según los conjuntos de reglas y los analizadores que especificó.

Métricas relacionadas: cuando se generan observaciones, la columna de métricas relacionadas muestra la regla y los valores reales y esperados, al igual que los límites inferior y superior.

Recomendaciones de reglas: a continuación, AWS Glue también recomienda reglas para tratar este problema. Cada regla recomendada se puede copiar haciendo clic en el icono de copia. Para copiar todas las reglas recomendadas, haga clic en el icono de copia situado junto a cada regla y, a continuación, en Aplicar las reglas copiadas.

Datos monitorizados: la columna de datos monitorizados presenta la columna o fila que se monitorizó y activó la observación.

Aplicar una regla de recomendación a su nodo de calidad de datos

Una vez que se haya generado una observación y se dé una regla recomendada, puede aplicarla a su nodo de calidad de datos. Para ello:

1. Haga clic en el icono de copia situado junto a cada recomendación de regla. Esto añadirá la recomendación de la regla a un bloc de notas que podrá consultar después.
2. Haga clic en Aplicar recomendaciones de reglas. Se abrirá el bloc de notas, donde podrá ver las reglas que ha copiado antes.
3. Elija Copiar conjunto de reglas.
4. Seleccione Aplicar al editor de conjuntos de reglas. Se abrirá el editor de conjuntos de reglas, donde puede pegar las reglas copiadas.
5. Pegue las reglas copiadas en el editor de conjuntos de reglas.

Calidad de datos para trabajos de ETL en las libretas AWS Glue Studio

En este tutorial, aprenderá a utilizar la calidad de datos de AWS Glue para trabajos de extracción, transformación y carga (ETL) en cuadernos de AWS Glue Studio.

Puede utilizar cuadernos en AWS Glue Studio para editar los guiones de los trabajos y ver el resultado sin tener que ejecutar un trabajo completo. También puede agregar anotaciones y guardar libretas como archivos `.ipynb` y guiones de trabajo. Tenga en cuenta que puede iniciar un cuaderno sin instalar software en forma local ni administrar servidores. Cuando esté satisfecho con el código, puede utilizar AWS Glue Studio para convertir fácilmente el cuaderno en un trabajo de AWS Glue.

El conjunto de datos que utiliza en este ejemplo está formado por datos de pago de Medicare Provider que se descargaron de dos sitios de Data.CMS.gov, conjuntos de datos: "Inpatient

Prospective Payment System Provider Summary for the Top 100 Diagnosis-Related Groups - FY2011" e "Inpatient Charge Data FY 2011".

Después de descargar los datos, modificamos el conjunto de datos para presentar un par de registros erróneos al final del archivo. Este archivo modificado se encuentra en un bucket público de Amazon S3 en `s3://awsglue-datasets/examples/medicare/Medicare_Hospital_Provider.csv`.

Requisitos previos

- Rol de AWS Glue con permiso de Amazon S3 para escribir en el bucket de Amazon S3 de destino
- Un cuaderno nuevo (consulte Introducción a los cuadernos en AWS Glue Studio)

Crear un trabajo de ETL en AWS Glue Studio

Para crear un trabajo de ETL

1. Cambie la versión de la sesión a AWS Glue 3.0.

   Para ello, elimine todas las celdas de código repetitivo con la siguiente magia y ejecute la celda. Tenga en cuenta que este código repetitivo se proporciona automáticamente en la primera celda cuando se crea un nuevo cuaderno.

   ```
   %glue_version 3.0
   ```

2. Copie el siguiente contenido y ejecútelo en la celda.

   ```
   import sys
   from awsglue.transforms import *
   from awsglue.utils import getResolvedOptions
   from pyspark.context import SparkContext
   from awsglue.context import GlueContext
   from awsglue.job import Job

   sc = SparkContext.getOrCreate()
   glueContext = GlueContext(sc)
   spark = glueContext.spark_session
   job = Job(glueContext)
   ```

3. En la celda siguiente, importe la clase de `EvaluateDataQuality` que evalúa la calidad de los datos de AWS Glue.

```
from awsgluedq.transforms import EvaluateDataQuality
```

4. En la celda siguiente, lea los datos de origen mediante el archivo.csv que está almacenado en el bucket público de Amazon S3.

```
medicare = spark.read.format(
"csv").option(
"header", "true").option(
"inferSchema", "true").load(
's3://awsglue-datasets/examples/medicare/Medicare_Hospital_Provider.csv')
medicare.printSchema()
```

5. Convierte los datos en un AWS Glue DynamicFrame.

```
from awsglue.dynamicframe import DynamicFrame
medicare_dyf = DynamicFrame.fromDF(medicare,glueContext,"medicare_dyf")
```

6. Cree el conjunto de reglas mediante el lenguaje de definición de calidad de datos (DQDL).

```
EvaluateDataQuality_ruleset = """
    Rules = [
        ColumnExists "Provider Id",
        IsComplete "Provider Id",
        ColumnValues  " Total Discharges " > 15
]
    ]
"""
```

7. Valide el conjunto de datos según el conjunto de reglas.

```
EvaluateDataQualityMultiframe = EvaluateDataQuality().process_rows(
    frame=medicare_dyf,
    ruleset=EvaluateDataQuality_ruleset,
    publishing_options={
        "dataQualityEvaluationContext": "EvaluateDataQualityMultiframe",
```

```
        "enableDataQualityCloudWatchMetrics": False,
        "enableDataQualityResultsPublishing": False,
    },
    additional_options={"performanceTuning.caching": "CACHE_NOTHING"},
)
```

8. Revise los resultados.

```
ruleOutcomes = SelectFromCollection.apply(
    dfc=EvaluateDataQualityMultiframe,
    key="ruleOutcomes",
    transformation_ctx="ruleOutcomes",
)

ruleOutcomes.toDF().show(truncate=False)
```

Salida:

```
-------------------------------------+-------
+-----------------------------------------------------
+-------------------------------------------+
|Rule                                 |Outcome|FailureReason
                  |EvaluatedMetrics                           |
+-------------------------------------+-------
+-----------------------------------------------------
+-------------------------------------------+
|ColumnExists "Provider Id"           |Passed |null
                  |{}                                         |
|IsComplete "Provider Id"             |Passed |null
                  |{Column.Provider Id.Completeness -> 1.0}   |
|ColumnValues " Total Discharges " > 15|Failed |Value: 11.0 does not meet the
 constraint requirement!|{Column. Total Discharges .Minimum -> 11.0}|
+-------------------------------------+-------
+-----------------------------------------------------
+-------------------------------------------+
```

9. Filtre las filas aprobadas y revise las filas fallidas a partir de los resultados según el nivel de fila de calidad de los datos.

```
owLevelOutcomes = SelectFromCollection.apply(
dfc=EvaluateDataQualityMultiframe,
key="rowLevelOutcomes",
transformation_ctx="rowLevelOutcomes",
)

rowLevelOutcomes_df = rowLevelOutcomes.toDF() # Convert Glue DynamicFrame to
 SparkSQL DataFrame
rowLevelOutcomes_df_passed =
 rowLevelOutcomes_df.filter(rowLevelOutcomes_df.DataQualityEvaluationResult ==
 "Passed") # Filter only the Passed records.
rowLevelOutcomes_df.filter(rowLevelOutcomes_df.DataQualityEvaluationResult ==
 "Failed").show(5, truncate=False) # Review the Failed records
```

Salida:

```
+----------------------------------------+-----------
+-------------------------------------+-------------------------+-------------
+--------------+-----------------+-----------------------------------
+------------------+-------------------------+----------------------
+-------------------------+-------------------------
+-----------------------------------------+---------------------------
+---------------------------+
|DRG Definition                           |Provider Id|Provider Name
           |Provider Street Address   |Provider City|Provider State|Provider Zip
 Code|Hospital Referral Region Description| Total Discharges | Average Covered
 Charges | Average Total Payments |Average Medicare Payments|DataQualityRulesPass
      |DataQualityRulesFail                    |DataQualityRulesSkip        |
DataQualityEvaluationResult|
+----------------------------------------+-----------
+-------------------------------------+-------------------------+-------------
+--------------+-----------------+-----------------------------------
+------------------+-------------------------+----------------------
+-------------------------+-------------------------
+-----------------------------------------+---------------------------
+---------------------------+
|039 - EXTRACRANIAL PROCEDURES W/O CC/MCC|10005      |MARSHALL MEDICAL CENTER SOUTH
        |2505 U S HIGHWAY 431 NORTH|BOAZ         |AL             |35957
 |AL - Birmingham                      |14               |$15131.85
  |$5787.57               |$4976.71                |[IsComplete "Provider Id"]|
```

```
[ColumnValues " Total Discharges " > 15]|[ColumnExists "Provider Id"]|Failed
              |
|039 - EXTRACRANIAL PROCEDURES W/O CC/MCC|10046      |RIVERVIEW REGIONAL MEDICAL
 CENTER    |600 SOUTH THIRD STREET    |GADSDEN      |AL            |35901
   |AL - Birmingham                      |14               |$67327.92
   |$5461.57               |$4493.57                |[IsComplete "Provider Id"]|
[ColumnValues " Total Discharges " > 15]|[ColumnExists "Provider Id"]|Failed
              |
|039 - EXTRACRANIAL PROCEDURES W/O CC/MCC|10083      |SOUTH BALDWIN REGIONAL
 MEDICAL CENTER|1613 NORTH MCKENZIE STREET|FOLEY        |AL            |36535
        |AL - Mobile                          |15               |$25411.33
         |$5282.93               |$4383.73                |[IsComplete "Provider
 Id"]|[ColumnValues " Total Discharges " > 15]|[ColumnExists "Provider Id"]|Failed
                    |
|039 - EXTRACRANIAL PROCEDURES W/O CC/MCC|30002      |BANNER GOOD SAMARITAN MEDICAL
 CENTER |1111 EAST MCDOWELL ROAD   |PHOENIX      |AZ            |85006
 |AZ - Phoenix                         |11               |$34803.81
  |$7768.90               |$6951.45                |[IsComplete "Provider Id"]|
[ColumnValues " Total Discharges " > 15]|[ColumnExists "Provider Id"]|Failed
              |
|039 - EXTRACRANIAL PROCEDURES W/O CC/MCC|30010      |CARONDELET ST  MARYS HOSPITAL
        |1601 WEST ST MARY'S ROAD  |TUCSON       |AZ            |85745
 |AZ - Tucson                          |12               |$35968.50
  |$6506.50               |$5379.83                |[IsComplete "Provider Id"]|
[ColumnValues " Total Discharges " > 15]|[ColumnExists "Provider Id"]|Failed
              |
+---------------------------------------+-----------
+-------------------------------------+-------------------------+-------------
+-------------+----------------+-----------------------------------
+-----------------+-------------------------+----------------------
+-------------------------+-------------------------
+---------------------------------------+--------------------------
+--------------------------+
only showing top 5 rows
```

Tenga en cuenta que AWS Glue Data Quality agregó cuatro columnas nuevas (DataQualityRulesPass DataQualityRulesFail DataQualityRulesSkip,, y DataQualityEvaluationResult). Esto indica los registros que se aprobaron, los registros que fallaron, las reglas omitidas para la evaluación a nivel de fila y los resultados generales a nivel de fila.

10. Escriba el resultado en un bucket de Amazon S3 para analizar los datos y visualizar los resultados.

```
#Write the Passed records to the destination.

glueContext.write_dynamic_frame.from_options(
       frame = rowLevelOutcomes_df_passed,
       connection_type = "s3",
       connection_options = {"path": "s3://glue-sample-target/output-dir/
medicare_parquet"},
       format = "parquet")
```

Referencia del lenguaje de definición de calidad de datos (DQDL)

El lenguaje de definición de calidad de datos (DQDL) es un lenguaje específico de dominio que se utiliza para definir las reglas de AWS Glue Data Quality.

En esta guía, se presentan los conceptos clave de DQDL para entender el lenguaje. También proporciona una referencia para los tipos de reglas de DQDL con sintaxis y ejemplos. Antes de utilizar esta guía, le recomendamos que se familiarice con AWS Glue Data Quality. Para obtener más información, consulte AWS Glue Calidad de los datos.

> **Note**
>
> DynamicRules solo se admiten en AWS Glue ETL.

Contenido

- Sintaxis de DQDL
  - Estructura de la regla
  - Reglas compuestas
    - Funcionamiento de las reglas compuestas
  - Expressions
    - Palabras clave para NULL, EMPTY y WHITESPACES_ONLY
    - Filtrar con la cláusula Where
  - Reglas dinámicas

- Analizadores
- Comentarios
- Referencia de tipo de regla de DQDL
  - AggregateMatch
  - ColumnCorrelation
  - ColumnCount
  - ColumnDataType
  - ColumnExists
  - ColumnLength
  - ColumnNamesMatchPattern
  - ColumnValues
  - Integridad
  - CustomSQL
  - DataFreshness
  - DatasetMatch
  - DistinctValuesCount
  - Entropía
  - IsComplete
  - IsPrimaryKey
  - IsUnique
  - Media
  - ReferentialIntegrity
  - RowCount
  - RowCountMatch
  - StandardDeviation
  - Sum
  - SchemaMatch
  - Singularidad
  - UniqueValueRatio
  - DetectAnomalies

Sintaxis de DQDL

Un documento de DQDL distingue entre mayúsculas y minúsculas y contiene un conjunto de reglas que agrupa las reglas individuales de la calidad de los datos. Para crear un conjunto de reglas, debe crear una lista denominada `Rules` (en mayúscula), delimitada por un par de corchetes. La lista debe contener una o más reglas de DQDL separadas por comas, como en el ejemplo siguiente.

```
Rules = [
   IsComplete "order-id",
   IsUnique "order-id"
]
```

Estructura de la regla

La estructura de una regla de DQDL depende del tipo de regla. Sin embargo, las reglas de DQDL se suelen ajustar al siguiente formato.

```
<RuleType> <Parameter> <Parameter> <Expression>
```

`RuleType` es el nombre que distingue entre mayúsculas y minúsculas del tipo de regla que quiere configurar. Por ejemplo, `IsComplete`, `IsUnique` o `CustomSql`. Los parámetros de la regla son diferentes para cada tipo de regla. Para obtener una referencia completa de los tipos de reglas de DQDL y sus parámetros, consulte Referencia de tipo de regla de DQDL.

Reglas compuestas

DQDL admite los siguientes operadores lógicos que puede utilizar para combinar las reglas. Estas reglas se denominan reglas compuestas.

y

El operador lógico `and` da como resultado `true`, siempre y cuando las reglas que conecta sean `true`. De lo contrario, la regla combinada da como resultado `false`. Cada regla que conecte con el operador `and` debe estar entre paréntesis.

El siguiente ejemplo usa el operador `and` para combinar dos reglas de DQDL.

```
(IsComplete "id") and (IsUnique "id")
```

o

El operador lógico `or` da como resultado `true`, siempre y cuando una o varias de las reglas que conecta sean `true`. Cada regla que conecte con el operador `or` debe estar entre paréntesis.

El siguiente ejemplo usa el operador `or` para combinar dos reglas de DQDL.

```
(RowCount "id" > 100) or (IsPrimaryKey "id")
```

Puede utilizar el mismo operador para conectar varias reglas, por lo que se permite la siguiente combinación de reglas.

```
(Mean "Star_Rating" > 3) and (Mean "Order_Total" > 500) and (IsComplete "Order_Id")
```

Sin embargo, no puede combinar los operadores lógicos en una sola expresión. Por ejemplo, no se permite la siguiente combinación.

```
(Mean "Star_Rating" > 3) and (Mean "Order_Total" > 500) or (IsComplete "Order_Id")
```

Funcionamiento de las reglas compuestas

De manera predeterminada, las reglas compuestas son evaluadas como reglas individuales a lo largo de todo el conjunto de datos y de la tabla y luego se combinan los resultados. En otras palabras, primero evalúan toda la columna y, a continuación, aplican el operador. Este comportamiento predeterminado se explica a continuación con un ejemplo:

```
# Dataset

+------+------+
|myCol1|myCol2|
+------+------+
|     2|     1|
|     0|     3|
+------+------+

# Overall outcome

+-----------------------------------------------------------+-------+
|Rule                                                       |Outcome|
```

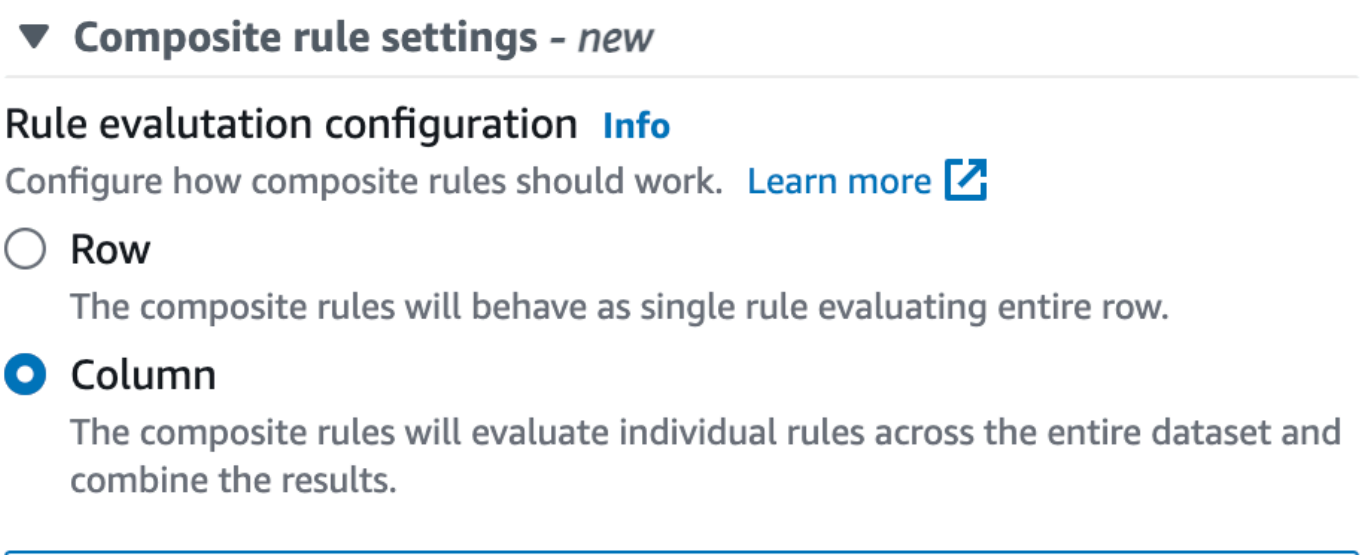
```
+-----------------------------------------------------------+-------+
|(ColumnValues "myCol1" > 1) OR (ColumnValues "myCol2" > 2)|Failed |
+-----------------------------------------------------------+-------+
```

En el ejemplo anterior, AWS Glue Data Quality primero evalúa `(ColumnValues "myCol1" > 1)`, lo cual resultará en una falla. Luego evaluará `(ColumnValues "myCol2" > 2)`, que también fallará. La combinación de los dos resultados se indicará como fallida.

Sin embargo, si prefiere SQL como el comportamiento en caso de que necesite evaluar una fila entera, tiene que establecer de manera explícita el parámetro `ruleEvaluation.scope` como se muestra en `additionalOptions`, en el fragmento de código a continuación.

```
object GlueApp {
  val datasource = glueContext.getCatalogSource(
    database="<db>",
    tableName="<table>",
    transformationContext="datasource"
  ).getDynamicFrame()

  val ruleset = """
    Rules = [
        (ColumnValues "age" >= 26) OR (ColumnLength "name" >= 4)
    ]
  """

  val dq_results = EvaluateDataQuality.processRows(
    frame=datasource,
    ruleset=ruleset,
    additionalOptions=JsonOptions("""
        {
          "compositeRuleEvaluation.method":"ROW"
        }
      """
    )
  )
}
```

En AWS Glue Studio y AWS Glue Data Catalog, puede configurar fácilmente esta opción en la interfaz de usuario, tal y como se muestra a continuación.



Una vez establecidas, las reglas compuestas se comportarán como una regla única que evalúa la fila en su totalidad. En el ejemplo a continuación se muestra este comportamiento.

```
# Row Level outcome

+------+------+-------------------------------------------------------------
+---------------------------+
|myCol1|myCol2|DataQualityRulesPass                                         |
DataQualityEvaluationResult|
+------+------+-------------------------------------------------------------
+---------------------------+
|2     |1     |[(ColumnValues "myCol1" > 1) OR (ColumnValues "myCol2" > 2)]|Passed
               |
|0     |3     |[(ColumnValues "myCol1" > 1) OR (ColumnValues "myCol2" > 2)]|Passed
               |
+------+------+-------------------------------------------------------------
+---------------------------+
```

Algunas reglas no son compatibles con esta característica debido a que los resultados generales se basan en umbrales o proporciones. Estas regulas se enumeran a continuación.

Reglas que se basan en las proporciones:

- Integridad

- DatasetMatch
- ReferentialIntegrity
- Singularidad

Reglas que se basan en los umbrales:

Cuando las reglas que se mencionan a continuación incluyen con un umbral, no son compatibles. Sin embargo, las reglas que no incluyen `with threshold` sí mantienen la compatibilidad.

- ColumnDataType
- ColumnValues
- CustomSQL

Expressions

Si un tipo de regla no produce una respuesta booleana, debe proporcionar una expresión como parámetro para crear una respuesta booleana. Por ejemplo, la siguiente regla compara la media (promedio) de todos los valores de una columna con una expresión para obtener un resultado true (verdadero) o false (falso).

```
Mean "colA" between 80 and 100
```

Algunos tipos de reglas, como `IsUnique` y `IsComplete` ya producen una respuesta booleana.

En la tabla siguiente, se enumeran las expresiones que puede utilizar en las reglas de DQDL.

Expresiones de DQDL admitidas

Expression	Descripción	Ejemplo
=*x*	Se resuelve en `true` si la respuesta del tipo de regla es igual que *x*.	`Completeness "colA" = "1.0", ColumnValues "colA" = "2022-06-30"`
!=*x*	Se resuelve en verdadero si la respuesta del tipo de regla es igual a *x*.	`ColumnValues "colA" != "a",`

Expression	Descripción	Ejemplo
		`ColumnValues "colA" != "2022-06-30"`
> *x*	Se resuelve en `true` si la respuesta del tipo de regla es mayor que *x*.	`ColumnValues "colA" > 10`
< *x*	Se resuelve en `true` si la respuesta del tipo de regla es menor que *x*.	`ColumnValues "colA" < 1000, ColumnValues "colA" < "2022-06-30"`
>= *x*	Se resuelve en `true` si la respuesta del tipo de regla es mayor o igual que *x*.	`ColumnValues "colA" >= 10`
<= *x*	Se resuelve en `true` si la respuesta del tipo de regla es menor o igual que *x*.	`ColumnValues "colA" <= 1000`
entre *x* e *y*	Se resuelve en `true` si la respuesta del tipo de regla se encuentra en un rango especificado (exclusivo). Solo debe utilizar este tipo de expresión para los tipos numéricos y de datos.	`Mean "colA" between 8 and 100, ColumnValues "colA" between "2022-05-31" and "2022-06-30"`
no está entre *x* e *y*	Se resuelve en verdadero si la respuesta del tipo de regla no se encuentra en un rango especificado (inclusivo). Solo debe utilizar este tipo de expresión para los tipos numéricos y de fecha.	`ColumnValues "colA" not between "2022-05-31" and "2022-06-30"`

Expression	Descripción	Ejemplo
en [*a, b, c, ...*]	Se resuelve en `true` si la respuesta del tipo de regla se encuentra en el conjunto especificado.	`ColumnValues "colA" in [ 1, 2, 3 ], ColumnValues "colA" in [ "a", "b", "c" ]`
no está en [*a, b, c,...*]	Se resuelve en `true` si la respuesta del tipo de regla no se encuentra en el conjunto especificado.	`ColumnValues "colA" not in [ 1, 2, 3 ], ColumnValues "colA" not in [ "a", "b", "c" ]`
coincide con */ab+c/i*	Se resuelve en `true` si la respuesta del tipo de regla coincide con una expresión regular.	`ColumnValues "colA" matches "[a-zA-Z]*"`
no coincide con */ab+c/i*	Se resuelve `true` si la respuesta del tipo de regla no coincide con una expresión regular.	`ColumnValues "colA" not matches "[a-zA-Z]*"`
now( )	Solo funciona con el tipo de regla `ColumnValues` para crear una expresión de fecha.	`ColumnValues "load_date" > (now() - 3 days)`
en/coincide con [...] / no está en/no coincide con [...] `with threshold`	Indica el porcentaje de valores que coinciden con las condiciones de la regla. Solo funciona con los tipos de regla `ColumnValues` , `ColumnDataType` y `CustomSQL` .	`ColumnValues "colA" in ["A", "B"] with threshold > 0.8, ColumnValues "colA" matches "[a-zA-Z]*" with threshold between 0.2 and 0.9 ColumnDataType "colA" = "Timestamp" with threshold > 0.9`

Palabras clave para NULL, EMPTY y WHITESPACES_ONLY

Para validar si una columna de cadenas contiene una cadena nula, vacía o solo con espacios en blanco, puede utilizar las palabras clave a continuación:

- NULL/null: esta palabra clave se resuelve como verdadera para un valor `null` en una columna de cadenas.

  `ColumnValues "colA" != NULL with threshold > 0.5` devolvería el valor verdadero si más del 50 % de los datos no tienen valores nulos.

  `(ColumnValues "colA" = NULL) or (ColumnLength "colA" > 5)` devolvería el valor verdadero para todas las filas que tengan un valor nulo o una longitud superior a 5. Tenga en cuenta que esto requerirá el uso de la opción «compositeRuleEvaluation.method» = «ROW».
- EMPTY/empty: esta palabra clave se resuelve como verdadera para un valor de cadena vacío (“”) en una columna de cadenas. Algunos formatos de datos transforman los valores nulos de una columna de cadenas en cadenas vacías. Esta palabra clave ayuda a remover las cadenas vacías en los datos.

  `(ColumnValues "colA" = EMPTY) or (ColumnValues "colA" in ["a", "b"])` devolvería el valor verdadero si una fila está vacía, “a” o “b”. Tenga en cuenta que esto requiere el uso de la opción «compositeRuleEvaluation.method» = «ROW».
- WHITESPACES_ONLY/whitespaces_only: esta palabra clave se resuelve como verdadera para una cadena que solo tiene un valor de espacios en blanco (“ ”) en una columna de cadenas.

  `ColumnValues "colA" not in ["a", "b", WHITESPACES_ONLY]` devolvería el valor verdadero si una fila no es ni “a” ni “b” ni solo contiene espacios en blanco.

  Reglas admitidas:

  - ColumnValues

Puede utilizar las palabras clave a continuación para validar si una columna contiene un valor nulo al trabajar con expresiones numéricas o basadas en fechas.

- NULL/null: esta palabra clave se resuelve como verdadera para un valor nulo en una columna de cadenas.

  `ColumnValues "colA" in [NULL, "2023-01-01"]` devolvería el valor verdadero si una fecha de la columna es `2023-01-01` o nula.

`(ColumnValues "colA" = NULL) or (ColumnValues "colA" between 1 and 9)` devolvería el valor verdadero para todas las filas que tengan un valor nulo o cuyos valores sean entre 1 y 9. Tenga en cuenta que esto requerirá el uso de la opción «compositeRuleEvaluation.method» = «ROW».

Reglas admitidas:

- ColumnValues

Filtrar con la cláusula Where

Puede filtrar sus datos al crear reglas. Esto resulta útil cuando se quieren aplicar reglas condicionales.

```
<DQDL Rule> where "<valid SparkSQL where clause> "
```

El filtro debe especificarse con la `where` palabra clave, seguida de una sentencia SparkSQL válida entre comillas. `("")`

Si desea añadir la cláusula where a una regla con un umbral, la cláusula where debe especificarse antes de la condición de umbral.

```
<DQDL Rule> where "valid SparkSQL statement>" with threshold <threshold condition>
```

Con esta sintaxis, puede escribir reglas como las siguientes.

```
Completeness "colA" > 0.5 where "colB = 10"
ColumnValues "colB" in ["A", "B"] where "colC is not null" with threshold > 0.9
ColumnLength "colC" > 10 where "colD != Concat(colE, colF)"
```

Validaremos que la sentencia SparkSQL proporcionada sea válida. Si no es válida, la evaluación de la regla fallará y la lanzaremos `IllegalArgumentException` con el siguiente formato:

```
Rule <DQDL Rule> where "<invalid SparkSQL>" has provided an invalid where clause :
<SparkSQL Error>
```

Comportamiento de la cláusula Where cuando la identificación del registro de errores a nivel de fila está activada

Con AWS Glue Data Quality, puede identificar registros específicos que fallaron. Al aplicar una cláusula where a las reglas que admiten resultados a nivel de fila, etiquetaremos las filas filtradas por la cláusula where como`Passed`.

Si prefiere etiquetar por separado las filas filtradas como`SKIPPED`, puede configurar lo siguiente `additionalOptions` para el trabajo de ETL.

```
object GlueApp {
  val datasource = glueContext.getCatalogSource(
    database="<db>",
    tableName="<table>",
    transformationContext="datasource"
  ).getDynamicFrame()

  val ruleset = """
    Rules = [
       IsComplete "att2" where "att1 = 'a'"
    ]
  """

  val dq_results = EvaluateDataQuality.processRows(
    frame=datasource,
    ruleset=ruleset,
    additionalOptions=JsonOptions("""
        {
          "rowLevelConfiguration.filteredRowLabel":"SKIPPED"
        }
      """
    )
  )
}
```

Como ejemplo, consulte la siguiente regla y marco de datos:

```
IsComplete att2 where "att1 = 'a'"
```

id	att1	att2	Resultados a nivel de fila (predeterminado)	Resultados a nivel de fila (opción omitida)	Comentarios
1	a	f	PASADO	PASADO	
2	b	d	PASADO	SKIPPED	La fila está filtrada, ya que no `att1` lo está `"a"`
3	a	null	ERROR	ERROR	
4	a	f	PASADO	PASADO	
5	b	null	PASADO	SKIPPED	La fila está filtrada, ya que no `att1` lo está `"a"`
6	a	f	PASADO	PASADO	

Reglas dinámicas

Ahora puede crear reglas dinámicas para comparar las métricas actuales generadas por sus reglas con sus valores históricos. Estas comparaciones históricas se habilitan usando el operador `last()` en las expresiones. Por ejemplo, la regla `RowCount > last()` se aplicará correctamente cuando el número de filas de la ejecución actual sea superior al recuento de filas anterior más reciente del mismo conjunto de datos. `last()` utiliza un argumento numérico natural opcional que describe cuántas métricas anteriores se deben tener en cuenta; `last(k)` donde `k >= 1` hará referencia a las últimas métricas `k`.

- Si no hay puntos de datos disponibles, `last(k)` devolverá el valor predeterminado, 0,0.
- Si hay menos de las métricas `k` disponibles, `last(k)` devolverá todas las métricas anteriores.

Para formar expresiones válidas utilice `last(k)`, donde `k > 1` requiera una función de agregación para reducir varios resultados históricos a un solo número. Por ejemplo, `RowCount > avg(last(5))` comprobará si el recuento de filas del conjunto de datos actual es estrictamente mayor que el promedio de los últimos cinco recuentos de filas del mismo conjunto de datos. `RowCount > last(5)` producirá un error, porque el recuento de filas del conjunto de datos actual no se puede comparar de forma significativa con una lista.

Funciones de agregación admitidas:

- `avg`
- `median`
- `max`
- `min`
- `sum`
- `std` (desviación estándar)
- `abs` (valor absoluto)
- `index(last(k), i)` permitirá seleccionar el valor en la posición `i` de los más recientes del último `k`. `i` está indexado a cero, por lo que `index(last(3), 0)` devolverá el punto de datos más reciente y `index(last(3), 3)` generará un error, ya que solo hay tres puntos de datos e intentaremos indexar el cuarto más reciente.

Ejemplos de expresiones

ColumnCorrelation

- `ColumnCorrelation "colA" "colB" < avg(last(10))`

DistinctValuesCount

- `DistinctValuesCount "colA" between min(last(10))-1 and max(last(10))+1`

La mayoría de los tipos de reglas con condiciones o umbrales numéricos admiten reglas dinámicas; consulte la tabla dada, Analizadores y reglas, para determinar si las reglas dinámicas son compatibles con su tipo de regla.

Analizadores

> **Note**
>
> El catálogo de datos de AWS Glue no admite analizadores.

Las reglas de DQDL usan funciones denominadas analizadores para recopilar información sobre los datos. La expresión booleana de una regla hace uso de esta información para determinar si la regla debe funcionar correctamente o no. Por ejemplo, la RowCount regla `RowCount > 5` utilizará un analizador de recuento de filas para descubrir el número de filas del conjunto de datos y compararlo con la expresión `> 5` para comprobar si existen más de cinco filas en el conjunto de datos actual.

A veces, en lugar de crear reglas, recomendamos crear analizadores y luego hacer que generen estadísticas que puedan usarse para detectar anomalías. Para estos casos, puede crear analizadores. Los analizadores se diferencian de las reglas en estas formas.

Característica	Analizadores	Reglas
Parte del conjunto de reglas	Sí	Sí
Genera estadísticas	Sí	Sí
Genera observaciones	Sí	Sí
Puede evaluar y hacer valer una condición	No	Sí
Puede configurar acciones como detener los trabajos por fallas o continuar procesando el trabajo	No	Sí

Los analizadores pueden existir de forma independiente sin reglas, por lo que es posible configurarlos rápidamente y crear progresivamente reglas de calidad de datos.

Algunos tipos de reglas se pueden introducir en el bloque `Analyzers` del conjunto de reglas para ejecutar las reglas necesarias para los analizadores y recopilar información sin comprobar ninguna

condición. Algunos analizadores no están asociados a las reglas y solo se pueden introducir en el bloque `Analyzers`. La siguiente tabla indica si cada elemento se admite como regla o como analizador independiente, junto con información adicional para cada tipo de regla.

Ejemplos de conjunto de reglas con el analizador

El siguiente conjunto de reglas usa:

- una regla dinámica para comprobar si un conjunto de datos está creciendo por arriba de su media final en las últimas tres ejecuciones de tareas,
- un analizador `DistinctValuesCount` para registrar la cantidad de valores distintos de la columna del conjunto de datos de `Name`,
- un analizador `ColumnLength` para rastrear el tamaño mínimo y máximo de `Name` a lo largo del tiempo.

Los resultados de las métricas del analizador pueden verse en la pestaña Calidad de los datos durante la ejecución del trabajo.

```
Rules = [
   RowCount > avg(last(3))
]
Analyzers = [
   DistinctValuesCount "Name",
   ColumnLength "Name"
]
```

Comentarios

Puede utilizar el carácter “#” para agregar un comentario al documento DQDL. DQDL ignorará todo lo que figure a continuación del “#” hasta el final de la línea.

```
Rules = [
    # More items should generally mean a higher price, so correlation should be
 positive
    ColumnCorrelation "price" "num_items" > 0
]
```

Referencia de tipo de regla de DQDL

Esta sección proporciona una referencia para cada tipo de regla compatible con AWS Glue Data Quality.

> **Note**
>
> - En la actualidad, DQDL no admite datos de columnas anidadas o de tipo lista.
> - Los valores entre corchetes de la siguiente tabla se sustituirán por la información dada en los argumentos de la regla.
> - Por lo general, las reglas requieren un argumento adicional para la expresión.

RuleType	Descripción	Arguments	Métricas informadas	¿Se admite como regla?	¿Se admite como analizador?	¿Devuelve los resultados de nivel de fila?	¿Soporte para reglas dinámicas?	Genera observaciones	¿Admite la sintaxis de la cláusula Where?
AggregateMatch	Compruebe si dos conjuntos de datos coinciden mediante la comparación de métricas resumidas, como el importe	Una o más agregaciones	Cuando los nombres de la primera y la segunda columna de agregación son los mismos:	Sí	No	No	No	No	No

RuleTyp	Descripc ón	Argumer s	Métricas informad s	¿Se adminte como regla?	¿Se admite como analizad r?	¿Devuel los resultad s de nivel de fila?	¿Soporte para reglas dinámica ?	Genera observac ones	¿Admite la sintaxis de la cláusula Where?

	total de ventas. Es útil para que las instituci ones financier as compare si todos los datos proviene de los sistemas de origen.		`Column [C olumn] gregate tch`
			
			Cuando los nombres de la primera y la segunda columna de agregaci n son distintos :
			
			`Column [C olumn1, lumn2] gregate tch`						

RuleTyp	Descripc ón	Argumer s	Métricas informad s	¿Se admite como regla?	¿Se admite como analizad r?	¿Devuel los resultad s de nivel de fila?	¿Soporte para reglas dinámica ?	Genera observac ones	¿Admite la sintaxis de la cláusula Where?

AllStatis tics	Analizad r independ ente que recopila varias métricas para la columna dada o para todas las columna de un conjunto de datos.	Un nombre de columna única, O "AllColur ns»	Para las columna de todos los tipos:
			`Datase .RowCo`
			`Column [C olumn]. mpleter s`
			`Column [C olumn]. iquene`
			Métricas adicional es para columna con valores
				No	Sí	No	No	No	No

RuleTyp	Descripc ón	Argumer s	Métricas informad s	¿Se adminte como regla?	¿Se admite como analizad r?	¿Devuel los resultad s de nivel de fila?	¿Soporte para reglas dinámica ?	Genera observac ones	¿Admite la sintaxis de la cláusula Where?

			de cadena:
			`ColumnL gth metrics`
			Métricas adicional es para columna con valores numéric :
			`Column\ ues metrics`						

ColumnC relation	Compru qué tan bien están correlaci onadas dos columna	Exactam te dos columna de nombres	`Multic mn. [Colum n1], [Colu mn2].Cc mnCorre tion`	Sí	Sí	No	Sí	No	Sí

RuleTyp	Descripc ón	Argumer s	Métricas informad s	¿Se admite como regla?	¿Se admite como analizad r?	¿Devuel los resultad s de nivel de fila?	¿Soporte para reglas dinámica ?	Genera observac ones	¿Admite la sintaxis de la cláusula Where?
ColumnC nt	Compru si se ha eliminad alguna columna	Ninguna	`Datase .Colum unt`	Sí	No	No	Sí	Sí	No
ColumnD aType	Compru si una columna es compatib e con un tipo de datos.	Exactam te un nombre de columna	`Column [C olumn]. lumnDat ype.Co iance`	Sí	No	No	Sí, en la expresió de umbral a nivel de fila	No	Sí

RuleTyp	Descripc ón	Argumer s	Métricas informad s	¿Se admite como regla?	¿Se admite como analizad r?	¿Devuel los resultad s de nivel de fila?	¿Soporte para reglas dinámica ?	Genera observac ones	¿Admite la sintaxis de la cláusula Where?
ColumnE sts	Compru si existen columna en un conjunto de datos. Esto les permite a los clientes crear plataform as de datos de autoserv cio para garantiza r que determin das columna estén	Exactam te un nombre de columna	N/A	Sí	No	No	No	No	No

RuleTyp	Descripc ón	Argumer s	Métricas informad s	¿Se adminte como regla?	¿Se admite como analizad r?	¿Devuel los resultad s de nivel de fila?	¿Soporte para reglas dinámica ?	Genera observac ones	¿Admite la sintaxis de la cláusula Where?
	disponibl es.								

ColumnL gth	Compru si la longitud de los datos es coherent .	Exactam te un nombre de columna	`Column [C olumn]. ximumLe th`
			
			`Column [C olumn]. nimumLe th`
			
			Métrica adicional cuando se da un umbral a nivel de fila:
			
			`Column [C olumn]. lumnVa s.Comp nce`
				Sí	Sí	Sí, cuando se da un umbral a nivel de fila	No	Sí. Solo genera observac ones analizan o la longitud mínima y máxima	Sí

RuleTyp	Descripc ón	Argumer s	Métricas informad s	¿Se adminte como regla?	¿Se admite como analizad r?	¿Devuel los resultad s de nivel de fila?	¿Soporte para reglas dinámica ?	Genera observac ones	¿Admite la sintaxis de la cláusula Where?
ColumnN esMatch ttern	Compru si los nombres de las columna coincide con patrones definidos . Es útil para que los equipos de gobierno refuerce la coheren a de los nombres de las columna	Una expresió regular para los nombres de las columna	`Datase .Colum mesPat nMatchF io`	Sí	No	No	No	No	No

RuleTyp	Descripc ón	Argumer s	Métricas informad s	¿Se admite como regla?	¿Se admite como analizad r?	¿Devuel los resultad s de nivel de fila?	¿Soporte para reglas dinámica ?	Genera observac ones	¿Admite la sintaxis de la cláusula Where?

ColumnV ues	Compru si los datos son coherent s según los valores definido . Esta regla admite expresio es regulare .	Exactam te un nombre de columna	`Column [C olumn] ximum`
			
			`Column [C olumn] nimum`
			
			Métrica adicional cuando se da un umbral a nivel de fila:
			
			`Column [C olumn] lumnVa s.Comp nce`
				Sí	Sí	Sí, cuando se da un umbral a nivel de fila	No	Sí. Solo genera observac ones analizan o los valores mínimo y máximo	Sí

RuleTyp	Descripc ón	Argumer s	Métricas informad s	¿Se adminte como regla?	¿Se admite como analizad r?	¿Devuel los resultad s de nivel de fila?	¿Soporte para reglas dinámica ?	Genera observac ones	¿Admite la sintaxis de la cláusula Where?
Integrida d	Compru si hay espacios en blanco o CEROS en los datos.	Exactam te un nombre de columna	`Column [C olumn] mpleter s`	Sí	Sí	Sí	Sí	Sí	Sí

CustomS	Los clientes pueden implem ar prácticar ente cualquie tipo de control de calidad de datos en SQL.	Una instrucci ón SQL
		
		(Opciona) Un umbral a nivel de fila
			`Dataset .Custor L`
			
			Métrica adicional cuando se da un umbral a nivel de fila:
			
			`Dataset .Custor L.Compl nce`
				Sí	No	Sí, cuando se da un umbral a nivel de fila	Sí	No	No

RuleTyp	Descripc ón	Argumer s	Métricas informad s	¿Se admite como regla?	¿Se admite como analizad r?	¿Devuel los resultad s de nivel de fila?	¿Soporte para reglas dinámica ?	Genera observac ones	¿Admite la sintaxis de la cláusula Where?
DataFres ness	Compru si los datos están actualiza dos.	Exactam te un nombre de columna	`Column [C olumn]. taFresh ss.Comp ance`	Sí	No	Sí	No	No	Sí
DatasetM tch	Compara dos conjunto de datos e identific a si están sincroniz ados.	Nombre de un conjunto de datos de referenci a Un mapeo de columna (Opciona) Columna para comprob si hay coincide cias	`Dataset [ Referer Dataset ias].Da setMat`	Sí	No	Sí	Sí	No	No

RuleTyp	Descripc ón	Argumer s	Métricas informad s	¿Se admite como regla?	¿Se admite como analizad r?	¿Devuel los resultad s de nivel de fila?	¿Soporte para reglas dinámica ?	Genera observac ones	¿Admite la sintaxis de la cláusula Where?
DistinctV aluesCo t	Compru si hay valores duplicad s.	Exactam te un nombre de columna	`Column [C olumn] stinct\ uesCou`	Sí	Sí	Sí	Sí	Sí	Sí
DetectAr malies	Compru la presenci de anomalí en las métricas notific as por otro tipo de regla.	Un tipo de regla	Métricas informad s por el argumen del tipo de regla	Sí	No	No	No	No	No
Entropía	Compru la entropía de los datos.	Exactam te un nombre de columna	`Column [C olumn] tropy`	Sí	Sí	No	Sí	No	Sí

RuleTyp	Descripc ón	Argumer s	Métricas informad s	¿Se adminte como regla?	¿Se admite como analizad r?	¿Devuel los resultad s de nivel de fila?	¿Soporte para reglas dinámica ?	Genera observac ones	¿Admite la sintaxis de la cláusula Where?
IsComple e	Compru si el 100 % de los datos está completc .	Exactam te un nombre de columna	`Column [C olumn]. mpleter s`	Sí	No	Sí	No	No	Sí
IsPrimar Key	Compru si una columna es una clave principal (no NULA y única).	Exactam te un nombre de columna	Para una sola columna `Column [C olumn]. iquenes` Para múltiples columna `Multic mn[Comm elimite olumns] niquene`	Sí	No	Sí	No	No	Sí

RuleTyp	Descripc ón	Argumer s	Métricas informad s	¿Se admite como regla?	¿Se admite como analizad r?	¿Devuel los resultad s de nivel de fila?	¿Soporte para reglas dinámica ?	Genera observac ones	¿Admite la sintaxis de la cláusula Where?
IsUnique	Compru si el 100 % de los datos es único.	Exactam te un nombre de columna	`Colum [C olumn] iquene`	Sí	No	Sí	No	No	Sí
Media	Compru si la media coincide con el umbral estableci do.	Exactam te un nombre de columna	`Colum [C olumn] an`	Sí	Sí	Sí	Sí	No	Sí

RuleTyp	Descripc ón	Argumer s	Métricas informad s	¿Se admite como regla?	¿Se admite como analizad r?	¿Devuel los resultad s de nivel de fila?	¿Soporte para reglas dinámica ?	Genera observac ones	¿Admite la sintaxis de la cláusula Where?
Referent alIntegri ty	Compru si dos conjunto de datos tienen integrida d referenci al.	Una o más columna del conjunto de datos Uno o más nombres de columna del conjunto de datos de referenci a	`Column [R eferen ataset/ as].Re ential egrity`	Sí	No	Sí	Sí	No	No
RowCou	Compru si los recuento de registros coincide con un umbral.	Ninguna	`Dataset .RowCou`	Sí	Sí	No	Sí	Sí	Sí

RuleTyp	Descripc ón	Argumer s	Métricas informad s	¿Se adminte como regla?	¿Se admite como analizad r?	¿Devuel los resultad s de nivel de fila?	¿Soporte para reglas dinámica ?	Genera observac ones	¿Admite la sintaxis de la cláusula Where?
RowCou atch	Comprue si los recuento de registros entre dos conjunto de datos coincide .	Alias del conjunto de datos de referenci a	`Dataset [ Referer Dataset ias].Rc ountMat`	Sí	No	No	Sí	No	No
Standarc eviation	Comprue si la desviaci n estándar coincide con el umbral.	Exactam te un nombre de columna	`Column [C olumn]. andardD iation`	Sí	Sí	Sí	Sí	No	Sí

RuleTyp	Descripc ón	Argumer s	Métricas informad s	¿Se adminte como regla?	¿Se admite como analizad r?	¿Devuel los resultad s de nivel de fila?	¿Soporte para reglas dinámica ?	Genera observac ones	¿Admite la sintaxis de la cláusula Where?
Schemal ch	Compru si el esquem entre dos conjunto de datos coincide.	Alias del conjunto de datos de referenci a	`Datase [ Refere Datase ias].Sc maMatc`	Sí	No	No	Sí	No	No
Sum	Compru si la suma coincid con un umbral estableci do.	Exactam te un nombre de columna	`Column [C olumn] m`	Sí	Sí	No	Sí	No	Sí
Singulari dad	Compru si la unicidad del conjunto de datos coincid con el umbral.	Exactam te un nombre de columna	`Column [C olumn] iquene`	Sí	Sí	Sí	Sí	No	Sí

RuleTyp	Descripc ón	Argumer s	Métricas informad s	¿Se adminte como regla?	¿Se admite como analizad r?	¿Devuel los resultad s de nivel de fila?	¿Soporte para reglas dinámica ?	Genera observac ones	¿Admite la sintaxis de la cláusula Where?
UniqueV ueRatio	Compru si la relación de valores únicos coincide con el umbral.	Exactam te un nombre de columna	`Column [C olumn]. iqueVa Ratio`	Sí	Sí	Sí	Sí	No	Sí

Temas

- AggregateMatch
- ColumnCorrelation
- ColumnCount
- ColumnDataType
- ColumnExists
- ColumnLength
- ColumnNamesMatchPattern
- ColumnValues
- Integridad
- CustomSQL
- DataFreshness
- DatasetMatch
- DistinctValuesCount

- Entropía
- IsComplete
- IsPrimaryKey
- IsUnique
- Media
- ReferentialIntegrity
- RowCount
- RowCountMatch
- StandardDeviation
- Sum
- SchemaMatch
- Singularidad
- UniqueValueRatio
- DetectAnomalies

AggregateMatch

Comprueba la relación de dos agregaciones de columnas con respecto a una expresión determinada. Este tipo de regla funciona en varios conjuntos de datos. Se evalúan las agregaciones de dos columnas y se obtiene una relación al dividir el resultado de la agregación de la primera columna por el resultado de la agregación de la segunda columna. La relación se comprueba con la expresión proporcionada para producir una respuesta booleana.

Sintaxis

Agregación de columnas

```
ColumnExists <AGG_OPERATION> (<OPTIONAL_REFERENCE_ALIAS>.<COL_NAME>)
```

- AGG_OPERATION: la operación que se debe utilizar para la agregación. `sum` y `avg` se admiten en la actualidad.

  Tipos de columnas compatibles: byte, decimal, doble, flotante, entero, largo, corto

- OPTIONAL_REFERENCE_ALIAS: este parámetro debe proporcionarse si la columna proviene de un conjunto de datos de referencia y no del conjunto de datos principal. Si utilizas esta regla en el catálogo de datos de AWS Glue, tu alias de referencia debe tener el formato "<database_name>. <table_name>. <column_name>

  Tipos de columnas compatibles: byte, decimal, doble, flotante, entero, largo, corto
- COL_NAME: el nombre de la columna que se va a agregar.

  Tipos de columnas compatibles: byte, decimal, doble, flotante, entero, largo, corto

Ejemplo: promedio

```
"avg(rating)"
```

Ejemplo: suma

```
"sum(amount)"
```

Ejemplo: promedio de la columna en el conjunto de datos de referencia

```
"avg(reference.rating)"
```

Regla

```
AggregateMatch <AGG_EXP_1> <AGG_EXP_2> <EXPRESSION>
```

- AGG_EXP_1: la agregación de la primera columna.

  Tipos de columnas compatibles: byte, decimal, doble, flotante, entero, largo, corto

  Tipos de columnas compatibles: byte, decimal, doble, flotante, entero, largo, corto
- AGG_EXP_2: la agregación de la segunda columna.

  Tipos de columnas compatibles: byte, decimal, doble, flotante, entero, largo, corto

  Tipos de columnas compatibles: byte, decimal, doble, flotante, entero, largo, corto
- EXPRESSION: una expresión que se ejecuta en función de la respuesta del tipo de regla para producir un valor booleano. Para obtener más información, consulte Expressions.

Ejemplo: coincidencia agregada mediante suma

La siguiente regla de ejemplo comprueba si la suma de los valores en la columna `amount` es exactamente igual a la suma de los valores en la columna `total_amount`.

```
AggregateMatch "sum(amount)" "sum(total_amount)" = 1.0
```

Ejemplo: coincidencia agregada mediante promedio

La siguiente regla de ejemplo comprueba si el promedio de los valores en la columna `ratings` es igual al 90 %, como mínimo, del promedio de los valores en la columna `ratings` en el conjunto de datos `reference`. El conjunto de datos de referencia se proporciona como origen de datos adicional en la experiencia ETL o el Catálogo de datos.

En AWS Glue ETL, puedes usar:

```
AggregateMatch "avg(ratings)" "avg(reference.ratings)" >= 0.9
```

En el catálogo de datos de AWS Glue, puede utilizar:

```
AggregateMatch "avg(ratings)" "avg(database_name.tablename.ratings)" >= 0.9
```

Comportamiento nulo

La regla `AggregateMatch` ignorará las filas con valores nulos al momento de calcular los métodos de agregación (suma/media). Por ejemplo:

```
+---+-----------+
|id |units      |
+---+-----------+
|100|0          |
|101|null       |
|102|20         |
|103|null       |
|104|40         |
+---+-----------+
```

La media de la columna `units` será (0 + 20 + 40) / 3 = 20. Las filas 101 y 103 no se tienen en cuenta para este cálculo.

ColumnCorrelation

Comprueba la correlación entre dos columnas y una expresión determinada. AWS Glue Data Quality utiliza el coeficiente de correlación de Pearson para medir la correlación lineal entre dos columnas. El resultado es un número entre -1 y 1 que mide la fuerza y la dirección de la relación.

Sintaxis

```
ColumnCorrelation <COL_1_NAME> <COL_2_NAME> <EXPRESSION>
```

- COL_1_NAME: el nombre de la primera columna con la que quiere evaluar la regla de la calidad de los datos.

  Tipos de columnas compatibles: byte, decimal, doble, flotante, entero, largo, corto
- COL_2_NAME: el nombre de la segunda columna con la que quiere evaluar la regla de la calidad de los datos.

  Tipos de columnas compatibles: byte, decimal, doble, flotante, entero, largo, corto
- EXPRESSION: una expresión que se ejecuta en función de la respuesta del tipo de regla para producir un valor booleano. Para obtener más información, consulte Expressions.

Ejemplo: correlación de columnas

La siguiente regla de ejemplo comprueba si el coeficiente de correlación entre las columnas `height` y `weight` tiene una correlación positiva fuerte (un valor de coeficiente superior a 0,8).

```
ColumnCorrelation "height" "weight" > 0.8
```

```
ColumnCorrelation "weightinkgs" "Salary" > 0.8 where "weightinkgs > 40"
```

Muestra de reglas dinámicas

- `ColumnCorrelation "colA" "colB" between min(last(10)) and max(last(10))`
- `ColumnCorrelation "colA" "colB" < avg(last(5)) + std(last(5))`

Comportamiento nulo

La regla `ColumnCorrelation` ignorará las filas con los valores `NULL` al momento de calcular la correlación. Por ejemplo:

```
+---+-----------+
|id |units      |
+---+-----------+
|100|0          |
|101|null       |
|102|20         |
|103|null       |
|104|40         |
+---+-----------+
```

Las filas 101 y 103 se ignorarán y la `ColumnCorrelation` será 1.0.

ColumnCount

Comprueba el recuento de columnas del conjunto de datos principal con respecto a una expresión determinada. En la expresión, puede indicar el número de columnas o un rango de columnas mediante operadores como > y <.

Sintaxis

```
ColumnCount <EXPRESSION>
```

- EXPRESSION: una expresión que se ejecuta en función de la respuesta del tipo de regla para producir un valor booleano. Para obtener más información, consulte Expressions.

Ejemplo: comprobación numérica del recuento de columnas

La siguiente regla de ejemplo comprueba si el recuento de columnas se encuentra dentro de un rango determinado.

```
ColumnCount between 10 and 20
```

Muestra de reglas dinámicas

- `ColumnCount >= avg(last(10))`
- `ColumnCount between min(last(10))-1 and max(last(10))+1`

ColumnDataType

Comprueba el tipo de datos inherente de los valores de una columna dada con respecto al tipo esperado proporcionado. Acepta una expresión `with threshold` para comprobar si hay un subconjunto de los valores en la columna.

Sintaxis

```
ColumnDataType <COL_NAME> = <EXPECTED_TYPE>
   ColumnDataType <COL_NAME> = <EXPECTED_TYPE> with threshold <EXPRESSION>
```

- COL_NAME: el nombre de la columna con la que quiere evaluar la regla de la calidad de los datos.

  Tipos de columnas compatibles: tipo cadena

  Tipos de columnas compatibles: byte, decimal, doble, flotante, entero, largo, corto
- EXPECTED_TYPE: el tipo esperado de los valores en la columna.

  Valores admitidos: booleano, fecha, marca de tiempo, entero, doble, flotante, largo

  Tipos de columnas compatibles: byte, decimal, doble, flotante, entero, largo, corto
- EXPRESIÓN: una expresión opcional para especificar el porcentaje de valores que deben ser del tipo esperado.

  Tipos de columnas compatibles: byte, decimal, doble, flotante, entero, largo, corto

Ejemplo: números enteros del tipo de datos de columna como cadenas

La siguiente regla de ejemplo comprueba si los valores de la columna dada, que es de tipo cadena, son en realidad enteros.

```
ColumnDataType "colA" = "INTEGER"
```

Ejemplo: los números enteros del tipo de datos de las columnas como cadenas comprueban un subconjunto de los valores

La siguiente regla de ejemplo comprueba si más del 90 % de los valores de la columna dada, que es de tipo cadena, son realmente enteros.

```
ColumnDataType "colA" = "INTEGER" with threshold > 0.9
```

ColumnExists

Comprueba si existe una columna.

Sintaxis

```
ColumnExists <COL_NAME>
```

- COL_NAME: el nombre de la columna con la que quiere evaluar la regla de la calidad de los datos.

  Tipos de columnas compatibles: cualquier tipo de columna

Ejemplo: la columna existe

La siguiente regla de ejemplo comprueba si existe la columna denominada `Middle_Name`.

```
ColumnExists "Middle_Name"
```

ColumnLength

Comprueba si la longitud de cada fila de una columna se ajusta a una expresión dada.

Sintaxis

```
ColumnLength <COL_NAME><EXPRESSION>
```

- COL_NAME: el nombre de la columna con la que quiere evaluar la regla de la calidad de los datos.

  Tipos de columnas compatibles: cadena
- EXPRESSION: una expresión que se ejecuta en función de la respuesta del tipo de regla para producir un valor booleano. Para obtener más información, consulte Expressions.

Ejemplo: longitud de la fila de la columna

La siguiente regla de ejemplo comprueba si el valor de cada fila de la columna denominada `Postal_Code` tiene 5 caracteres.

```
ColumnLength "Postal_Code" = 5
ColumnLength "weightinkgs" = 2 where "weightinkgs > 10"
```

Comportamiento nulo

La regla `ColumnLength` considera a los valores `NULL` como cadenas con una longitud de 0. Para una fila `NULL`:

```
ColumnLength "Postal_Code" > 4 # this will fail
```

```
ColumnLength "Postal_Code" < 6 # this will succeed
```

El ejemplo de regla compuesta a continuación presenta una manera de fallar de manera explícita un valor `NULL`:

```
(ColumnLength "Postal_Code" > 4) AND (ColumnValues != NULL)
```

ColumnNamesMatchPattern

Comprueba si los nombres de todas las columnas del conjunto de datos principal coinciden con la expresión regular dada.

Sintaxis

```
ColumnNamesMatchPattern <PATTERN>
```

- PATRÓN: el patrón contra el que desea evaluar la regla de calidad de los datos.

  Tipos de columnas compatibles: byte, decimal, doble, flotante, entero, largo, corto

Ejemplo: los nombres de las columnas coinciden con el patrón

La siguiente regla de ejemplo comprueba si todas las columnas comienzan con el prefijo “aws_”

```
ColumnNamesMatchPattern "aws_.*"
ColumnNamesMatchPattern "aws_.*" where "weightinkgs > 10"
```

ColumnValues

Ejecuta una expresión con los valores de una columna.

Sintaxis

```
ColumnValues <COL_NAME> <EXPRESSION>
```

- COL_NAME: el nombre de la columna con la que quiere evaluar la regla de la calidad de los datos.

  Tipos de columnas compatibles: cualquier tipo de columna
- EXPRESSION: una expresión que se ejecuta en función de la respuesta del tipo de regla para producir un valor booleano. Para obtener más información, consulte Expressions.

Ejemplo: valores permitidos

El ejemplo de regla a continuación controla que cada valor en la columna específica se encuentre en un conjunto de valores permitidos (incluidos los valores nulos y las cadenas que solo contienen espacios en blanco).

```
ColumnValues "Country" in [ "US", "CA", "UK", NULL, EMPTY, WHITESPACES_ONLY ]
ColumnValues "gender" in ["F", "M"] where "weightinkgs < 10"
```

Ejemplo: expresión regular

La siguiente regla de ejemplo compara los valores de una columna con una expresión regular.

```
ColumnValues "First_Name" matches "[a-zA-Z]*"
```

Ejemplo: valores de fecha

La siguiente regla de ejemplo compara los valores de una columna de fecha con una expresión de fecha.

```
ColumnValues "Load_Date" > (now() - 3 days)
```

Ejemplo: valores numéricos

La siguiente regla de ejemplo comprueba si los valores de las columnas coinciden con una restricción numérica determinada.

```
ColumnValues "Customer_ID" between 1 and 2000
```

Comportamiento nulo

Para todas las reglas `ColumnValues` (además de `!=` y `NOT IN`), las filas `NULL` fallarán la regla. Si la regla falla debido a un valor nulo, la razón de la falla se mostrará de la siguiente manera:

```
Value: NULL does not meet the constraint requirement!
```

El ejemplo de regla compuesta a continuación presenta una manera de permitir de manera explícita los valores `NULL`:

```
(ColumnValues "Age" > 21) OR (ColumnValues "Age" = NULL)
```

ColumnValues Las reglas negadas que utilicen la `not in` sintaxis `!=` y se utilizarán para `NULL` las filas. Por ejemplo:

```
ColumnValues "Age" != 21
```

```
ColumnValues "Age" not in [21, 22, 23]
```

Los ejemplos a continuación presentan una manera de fallar de manera explícita los valores `NULL`

```
(ColumnValues "Age" != 21) AND (ColumnValues "Age" != NULL)
```

```
ColumnValues "Age" not in [21, 22, 23, NULL]
```

Integridad

Comprueba el porcentaje de valores completos (no nulos) de una columna con respecto a una expresión determinada.

Sintaxis

```
Completeness <COL_NAME> <EXPRESSION>
```

- COL_NAME: el nombre de la columna con la que quiere evaluar la regla de la calidad de los datos.

Tipos de columnas compatibles: cualquier tipo de columna

- EXPRESSION: una expresión que se ejecuta en función de la respuesta del tipo de regla para producir un valor booleano. Para obtener más información, consulte Expressions.

Ejemplo: porcentaje de valor nulo

Las siguientes reglas de ejemplo comprueban si más del 95 % de los valores de una columna están completos.

```
Completeness "First_Name" > 0.95
Completeness "First_Name" > 0.95 where "weightinkgs > 10"
```

Muestra de reglas dinámicas

- `Completeness "colA" between min(last(5)) - 1 and max(last(5)) + 1`
- `Completeness "colA" <= avg(last(10))`

Comportamiento nulo

Nota sobre los formatos de datos CSV: las filas en blanco en las columnas de CSV pueden demostrar distintos comportamientos.

- Si una columna es del tipo `String`, la fila en blanco se reconocerá como una cadena vacía y no fallará la regla `Completeness`.
- Si una columna es de otro tipo de dato, como `Int`, la fila en blanco se reconocerá como `NULL` y fallará la regla `Completeness`.

CustomSQL

Este tipo de regla se ha ampliado para admitir dos casos de uso:

- Ejecutar una instrucción SQL personalizada contra un conjunto de datos y comprobar el valor que se produce con respecto a una expresión dada.
- Ejecutar una instrucción SQL personalizada en la que especifique un nombre de columna en la instrucción SELECT contra el que compararlo con alguna condición para obtener resultados a nivel de fila.

Sintaxis

```
CustomSql <SQL_STATEMENT> <EXPRESSION>
```

- SQL_STATEMENT: una instrucción SQL que devuelve un solo valor numérico, entre comillas dobles.
- EXPRESSION: una expresión que se ejecuta en función de la respuesta del tipo de regla para producir un valor booleano. Para obtener más información, consulte Expressions.

Ejemplo: SQL personalizado para recuperar el resultado general de una regla

Esta regla de ejemplo usa una instrucción SQL para recuperar el recuento de registros de un conjunto de datos. A continuación, la regla comprueba que el recuento de registros esté comprendido entre 10 y 20.

```
CustomSql "select count(*) from primary" between 10 and 20
```

Ejemplo: SQL personalizado para recuperar los resultados a nivel de fila

Esta regla de ejemplo utiliza una instrucción SQL en la que se especifica un nombre de columna en la instrucción SELECT con el que se compara con alguna condición para obtener resultados a nivel de fila. Una expresión de condición de umbral define un umbral del número de registros que deben fallar para que falle toda la regla. Tenga en cuenta que una regla no puede contener una condición y una palabra clave juntas.

```
CustomSql "select Name from primary where Age  > 18"
```

o

```
CustomSql "select Name from primary where Age > 18" with threshold  > 3
```

> **Important**
>
> El alias `primary` representa el nombre del conjunto de datos que quiere evaluar. Al trabajar con trabajos de ETL visuales en la consola, `primary` siempre representa el elemento `DynamicFrame` que se pasa a la transformación de `EvaluateDataQuality.apply()`.

> Cuando utilizas el catálogo de datos de AWS Glue para ejecutar tareas de calidad de datos en una tabla, `primary` representa la tabla.

Si está en el Catálogo de datos de AWS Glue, también puede usar los nombres reales de las tablas:

```
CustomSql "select count(*) from database.table" between 10 and 20
```

También puede combinar varias tablas para comparar diferentes elementos de datos:

```
CustomSql "select count(*) from database.table inner join database.table2 on id1 = id2"
 between 10 and 20
```

En ETL de AWS Glue, CustomSQL puede identificar los registros que no superaron las comprobaciones de calidad de datos. Para que esto funcione, tendrá que producir los registros que forman parte de la tabla principal en la que evalúa la calidad de los datos. Los registros que se producen como parte de la consulta se consideran correctos y aquellos que no se producen se consideran fallidos.

La siguiente regla garantizará que los registros con una antigüedad inferior a 100 años se identifiquen como correctos y que aquellos anteriores se marquen como fallidos.

```
CustomSql "select id from primary where age < 100"
```

Esta regla CustomSQL se aprobará cuando el 50 % de los registros tengan más 10 años de antigüedad y también identificará los registros que fallaron. Los registros que este CustomSQL produce se considerarán aprobados, mientras que los que no se producen se considerarán fallidos.

```
CustomSQL "select ID, CustomerID from primary where age > 10" with threshold > 0.5
```

Nota: La regla CustomSQL fallará si usted produce registros que no están disponibles en el conjunto de datos.

DataFreshness

Comprueba la actualización de los datos de una columna mediante la evaluación de la diferencia entre la hora actual y los valores de una columna de fecha. Puede indicar una expresión basada

en el tiempo para este tipo de regla para asegurarse de que los valores de las columnas estén actualizados.

Sintaxis

```
DataFreshness <COL_NAME> <EXPRESSION>
```

- COL_NAME: el nombre de la columna con la que quiere evaluar la regla de la calidad de los datos.

  Tipos de columnas compatibles: fecha
- EXPRESSION: una expresión numérica en horas o días. Debe indicar la unidad de tiempo en la expresión.

Ejemplo: actualización de los datos

Las siguientes reglas de ejemplo comprueban la actualización de los datos.

```
DataFreshness "Order_Date" <= 24 hours
DataFreshness "Order_Date" between 2 days and 5 days
```

Comportamiento nulo

Las reglas `DataFreshness` fallarán para las filas con valores `NULL`. Si la regla falla debido a un valor nulo, la razón de la falla se mostrará de la siguiente manera:

```
80.00 % of rows passed the threshold
```

donde el 20 % de las filas que fallaron incluyen las filas con valores `NULL`.

El ejemplo de regla compuesta a continuación presenta una manera de permitir de manera explícita los valores `NULL`:

```
(DataFreshness "Order_Date" <= 24 hours) OR (ColumnValues "Order_Date" = NULL)
```

Actualización de datos para objetos de Amazon S3

En ocasiones, tendrá que validar la actualización de los datos en función del tiempo de creación del archivo de Amazon S3. Para ello, puede usar el siguiente código para obtener la marca temporal y

añadirla a su marco de datos y, a continuación, aplicar las comprobaciones de actualización de los datos.

```
df = glueContext.create_data_frame.from_catalog(database = "default", table_name =
 "mytable")
df = df.withColumn("file_ts", df["_metadata.file_modification_time"])

Rules = [
 DataFreshness "file_ts" < 24 hours
]
```

DatasetMatch

Compruebe si los datos del conjunto de datos principal coinciden con los datos de un conjunto de datos de referencia. Los dos conjuntos de datos se combinan mediante las asignaciones de columnas clave proporcionadas. Se pueden proporcionar asignaciones de columnas adicionales si desea comprobar la igualdad de los datos solo en esas columnas. Tenga en cuenta que, DataSetMatchpara que funcionen, las claves de unión deben ser únicas y no nulas (debe ser una clave principal). Si no cumple estas condiciones, recibirá el siguiente mensaje de error: “La asignación de teclas proporcionada no es adecuada para determinados marcos de datos”. En los casos en los que no puedas tener claves combinadas que sean únicas, considera usar otros tipos de reglas, como las que coincidan en AggregateMatchlos datos resumidos.

Sintaxis

```
DatasetMatch <REFERENCE_DATASET_ALIAS> <JOIN CONDITION WITH
 MAPPING> <OPTIONAL_MATCH_COLUMN_MAPPINGS> <EXPRESSION>
```

- REFERENCE_DATASET_ALIAS: el alias del conjunto de datos de referencia con el que se comparan los datos del conjunto de datos principal.
- KEY_COLUMN_MAPPINGS: una lista de nombres de columnas separados por comas que forman una clave en los conjuntos de datos. Si los nombres de las columnas no son los mismos en ambos conjuntos de datos, debe separarlos con un `->`
- OPTIONAL_MATCH_COLUMN_MAPPINGS: puede proporcionar este parámetro si desea comprobar si hay datos coincidentes solo en determinadas columnas. Utiliza la misma sintaxis que las asignaciones de columnas clave. Si no se proporciona este parámetro, haremos coincidir los datos en todas las columnas restantes. Las columnas restantes, que no son clave, deben tener los mismos nombres en ambos conjuntos de datos.

- EXPRESSION: una expresión que se ejecuta en función de la respuesta del tipo de regla para producir un valor booleano. Para obtener más información, consulte Expressions.

Ejemplo: haga coincidir los conjuntos de datos mediante la columna de ID

La siguiente regla de ejemplo comprueba que más del 90 % del conjunto de datos principal coincide con el conjunto de datos de referencia, mediante el uso de la columna “ID” para unir los dos conjuntos de datos. En este caso, compara todas las columnas.

```
DatasetMatch "reference" "ID" >= 0.9
```

Ejemplo: haga coincidir conjuntos de datos mediante varias columnas clave

En el siguiente ejemplo, el conjunto de datos principal y el conjunto de datos de referencia tienen nombres diferentes para las columnas clave. `ID_1` y `ID_2` juntos forman una clave compuesta en el conjunto de datos principal. `ID_ref1` y `ID_ref2` juntos forman una clave compuesta en el conjunto de datos de referencia. En este escenario, puede usar la sintaxis especial para proporcionar los nombres de las columnas.

```
DatasetMatch "reference" "ID_1->ID_ref1,ID_ref2->ID_ref2" >= 0.9
```

Ejemplo: haga coincidir conjuntos de datos mediante varias columnas clave y compruebe que una columna específica coincida

Este ejemplo se basa en el ejemplo anterior. Queremos comprobar que solo coincide la columna que contiene los importes. Esta columna se denomina `Amount1` en el conjunto de datos principal y `Amount2` en el conjunto de datos de referencia. Quiere una coincidencia exacta.

```
DatasetMatch "reference" "ID_1->ID_ref1,ID_ref2->ID_ref2" "Amount1->Amount2" >= 0.9
```

DistinctValuesCount

Compara la cantidad de valores distintos de una columna con una expresión dada.

Sintaxis

```
DistinctValuesCount <COL_NAME> <EXPRESSION>
```

- COL_NAME: el nombre de la columna con la que quiere evaluar la regla de la calidad de los datos.

Tipos de columnas compatibles: cualquier tipo de columna

- EXPRESSION: una expresión que se ejecuta en función de la respuesta del tipo de regla para producir un valor booleano. Para obtener más información, consulte Expressions.

Ejemplo: recuento de valores distintos de columnas

La siguiente regla de ejemplo comprueba que la columna denominada `State` contenga más de 3 valores distintos.

```
DistinctValuesCount "State" > 3
DistinctValuesCount "Customer_ID" < 6  where "Customer_ID < 10"
```

Muestra de reglas dinámicas

- `DistinctValuesCount "colA" between avg(last(10))-1 and avg(last(10))+1`
- `DistinctValuesCount "colA" <= index(last(10),2) + std(last(5))`

Entropía

Comprueba si el valor de entropía de una columna coincide con una expresión dada. La entropía mide el nivel de información que contiene un mensaje. Dada la distribución de probabilidad entre los valores de una columna, la entropía describe cuántos bits se necesitan para identificar un valor.

Sintaxis

```
Entropy <COL_NAME> <EXPRESSION>
```

- COL_NAME: el nombre de la columna con la que quiere evaluar la regla de la calidad de los datos.

  Tipos de columnas compatibles: cualquier tipo de columna
- EXPRESSION: una expresión que se ejecuta en función de la respuesta del tipo de regla para producir un valor booleano. Para obtener más información, consulte Expressions.

Ejemplo: entropía de columna

La siguiente regla de ejemplo comprueba que la columna denominada `Feedback` tiene un valor de entropía superior a uno.

```
Entropy "Star_Rating" > 1
Entropy "First_Name" > 1  where "Customer_ID < 10"
```

Muestra de reglas dinámicas

- `Entropy "colA" < max(last(10))`
- `Entropy "colA" between min(last(10)) and max(last(10))`

IsComplete

Comprueba si todos los valores de una columna están completos (no nulos).

Sintaxis

```
IsComplete <COL_NAME>
```

- COL_NAME: el nombre de la columna con la que quiere evaluar la regla de la calidad de los datos.

  Tipos de columnas compatibles: cualquier tipo de columna

Ejemplo: valores nulos

El siguiente ejemplo comprueba si todos los valores de una columna denominada `email` no son nulos.

```
IsComplete "email"
IsComplete "Email" where "Customer_ID between 1 and 50"
IsComplete "Customer_ID"  where "Customer_ID < 16 and Customer_ID != 12"
IsComplete "passenger_count" where "payment_type<>0"
```

Comportamiento nulo

Nota sobre los formatos de datos CSV: las filas en blanco en las columnas de CSV pueden demostrar distintos comportamientos.

- Si una columna es del tipo `String`, la fila en blanco se reconocerá como una cadena vacía y no fallará la regla `Completeness`.

- Si una columna es de otro tipo de dato, como `Int`, la fila en blanco se reconocerá como `NULL` y fallará la regla `Completeness`.

IsPrimaryKey

Comprueba si una columna contiene una clave principal. Una columna contiene una clave principal si todos los valores de la columna son únicos y completos (no nulos).

Sintaxis

```
IsPrimaryKey <COL_NAME>
```

- COL_NAME: el nombre de la columna con la que quiere evaluar la regla de la calidad de los datos.

  Tipos de columnas compatibles: cualquier tipo de columna

Ejemplo: clave principal

La siguiente regla de ejemplo comprueba si la columna denominada `Customer_ID` contiene una clave principal.

```
IsPrimaryKey "Customer_ID"
IsPrimaryKey "Customer_ID" where "Customer_ID < 10"
```

Ejemplo: clave principal con múltiples columnas. Cualquiera de los ejemplos siguientes son válidos.

```
IsPrimaryKey "colA" "colB"
IsPrimaryKey "colA" "colB" "colC"
IsPrimaryKey colA "colB" "colC"
```

IsUnique

Comprueba si todos los valores de una columna son únicos y devuelve un valor booleano.

Sintaxis

```
IsUnique <COL_NAME>
```

- COL_NAME: el nombre de la columna con la que quiere evaluar la regla de la calidad de los datos.

  Tipos de columnas compatibles: cualquier tipo de columna

Ejemplo: valores de columna únicos

La siguiente regla de ejemplo comprueba si todos los valores de una columna denominada `email` son únicos.

```
IsUnique "email"
IsUnique "Customer_ID" where "Customer_ID < 10"]
```

Media

Comprueba si la media (promedio) de todos los valores de una columna coincide con una expresión dada.

Sintaxis

```
Mean <COL_NAME> <EXPRESSION>
```

- COL_NAME: el nombre de la columna con la que quiere evaluar la regla de la calidad de los datos.

  Tipos de columnas compatibles: byte, decimal, doble, flotante, entero, largo, corto
- EXPRESSION: una expresión que se ejecuta en función de la respuesta del tipo de regla para producir un valor booleano. Para obtener más información, consulte Expressions.

Ejemplo: valor promedio

La siguiente regla de ejemplo comprueba si el promedio de todos los valores de una columna supera un umbral.

```
Mean "Star_Rating" > 3
Mean "Salary" < 6200 where "Customer_ID < 10"
```

Muestra de reglas dinámicas

- `Mean "colA" > avg(last(10)) + std(last(2))`

- `Mean "colA" between min(last(5)) - 1 and max(last(5)) + 1`

Comportamiento nulo

La regla `Mean` ignorará las filas con valores `NULL` al momento de calcular la media. Por ejemplo:

```
+---+-----------+
|id |units      |
+---+-----------+
|100|0          |
|101|null       |
|102|20         |
|103|null       |
|104|40         |
+---+-----------+
```

La media de la columna `units` será (0 + 20 + 40) / 3 = 20. Las filas 101 y 103 no se tienen en cuenta para este cálculo.

ReferentialIntegrity

Comprueba en qué medida los valores de un conjunto de columnas en el conjunto de datos principal son un subconjunto de los valores de un conjunto de columnas en un conjunto de datos de referencia.

Sintaxis

```
ReferentialIntegrity <PRIMARY_COLS> <REFERENCE_DATASET_COLS> <EXPRESSION>
```

- PRIMARY_COLS: una lista de nombres de columnas separados por comas en el conjunto de datos principal.

  Tipos de columnas compatibles: byte, decimal, doble, flotante, entero, largo, corto
- REFERENCE_DATASET_COLS: este parámetro contiene dos partes separadas por un punto. La primera parte es el alias del conjunto de datos de referencia. La segunda parte es la lista de nombres de columnas separados por comas en el conjunto de datos de referencia entre corchetes.

  Tipos de columnas compatibles: byte, decimal, doble, flotante, entero, largo, corto
- EXPRESSION: una expresión que se ejecuta en función de la respuesta del tipo de regla para producir un valor booleano. Para obtener más información, consulte Expressions.

Ejemplo: compruebe la integridad referencial de una columna de código postal

La siguiente regla de ejemplo comprueba que más del 90 % de los valores en la columna `zipcode` en el conjunto de datos principal estén presentes en la columna `zipcode` en el conjunto de datos de `reference`.

```
ReferentialIntegrity "zipcode" "reference.zipcode" >= 0.9
```

Ejemplo: compruebe la integridad referencial de las columnas de ciudad y estado

En el siguiente ejemplo, las columnas que contienen información sobre la ciudad y el estado existen en el conjunto de datos principal y en el conjunto de datos de referencia. Los nombres de las columnas son diferentes en ambos conjuntos de datos. La regla comprueba si el conjunto de valores de las columnas en el conjunto de datos principal es exactamente igual al conjunto de valores de las columnas en el conjunto de datos de referencia.

```
ReferentialIntegrity "city,state" "reference.{ref_city,ref_state}" = 1.0
```

Muestra de reglas dinámicas

- `ReferentialIntegrity "city,state" "reference.{ref_city,ref_state}" > avg(last(10))`
- `ReferentialIntegrity "city,state" "reference.{ref_city,ref_state}" between min(last(10)) - 1 and max(last(10)) + 1`

RowCount

Compara el recuento de filas de un conjunto de datos con una expresión determinada. En la expresión, puede indicar el número de filas o un rango de filas mediante operadores como > y <.

Sintaxis

```
RowCount <EXPRESSION>
```

- EXPRESSION: una expresión que se ejecuta en función de la respuesta del tipo de regla para producir un valor booleano. Para obtener más información, consulte Expressions.

Ejemplo: comprobación numérica del recuento de filas

La siguiente regla de ejemplo comprueba si el recuento de filas se encuentra dentro de un rango determinado.

```
RowCount between 10 and 100
RowCount between 1 and 50 where "Customer_ID < 10"
```

Muestra de reglas dinámicas

```
RowCount > avg(lats(10)) *0.8
```

RowCountMatch

Comprueba la relación entre el recuento de filas del conjunto de datos principal y el recuento de filas de un conjunto de datos de referencia con respecto a la expresión dada.

Sintaxis

```
RowCountMatch <REFERENCE_DATASET_ALIAS> <EXPRESSION>
```

- REFERENCE_DATASET_ALIAS: el alias del conjunto de datos de referencia con el que se comparan los recuentos de filas.

  Tipos de columnas compatibles: byte, decimal, doble, flotante, entero, largo, corto
- EXPRESSION: una expresión que se ejecuta en función de la respuesta del tipo de regla para producir un valor booleano. Para obtener más información, consulte Expressions.

Ejemplo: comparación del recuento de filas con un conjunto de datos de referencia

La siguiente regla de ejemplo comprueba si el recuento de filas del conjunto de datos principal es al menos el 90 % del recuento de filas del conjunto de datos de referencia.

```
RowCountMatch "reference" >= 0.9
```

StandardDeviation

Comprueba la desviación estándar de todos los valores de una columna con respecto a una expresión determinada.

Sintaxis

```
StandardDeviation <COL_NAME> <EXPRESSION>
```

- COL_NAME: el nombre de la columna con la que quiere evaluar la regla de la calidad de los datos.

  Tipos de columnas compatibles: byte, decimal, doble, flotante, entero, largo, corto
- EXPRESSION: una expresión que se ejecuta en función de la respuesta del tipo de regla para producir un valor booleano. Para obtener más información, consulte Expressions.

Ejemplo: desviación estándar

La siguiente regla de ejemplo comprueba si la desviación estándar de los valores de una columna denominada `colA` es inferior a un valor especificado.

```
StandardDeviation "Star_Rating" < 1.5
StandardDeviation "Salary" < 3500 where "Customer_ID < 10"
```

Muestra de reglas dinámicas

- `StandardDeviation "colA" > avg(last(10) + 0.1`
- `StandardDeviation "colA" between min(last(10)) - 1 and max(last(10)) + 1`

Comportamiento nulo

La regla `StandardDeviation` ignorará las filas con valores `NULL` al momento de calcular la desviación estándar. Por ejemplo:

```
+---+-----------+-----------+
|id |units1     |units2     |
+---+-----------+-----------+
|100|0          |0          |
|101|null       |0          |
|102|20         |20         |
|103|null       |0          |
|104|40         |40         |
+---+-----------+-----------+
```

La desviación estándar de la columna `units1` no tendrá en cuenta las filas 101 y 103 y su resultado será 16.33. La desviación estándar de la columna `units2` dará como resultado 16.

Sum

Comprueba la suma de todos los valores de una columna con respecto a una expresión determinada.

Sintaxis

```
Sum <COL_NAME> <EXPRESSION>
```

- COL_NAME: el nombre de la columna con la que quiere evaluar la regla de la calidad de los datos.

  Tipos de columnas compatibles: byte, decimal, doble, flotante, entero, largo, corto
- EXPRESSION: una expresión que se ejecuta en función de la respuesta del tipo de regla para producir un valor booleano. Para obtener más información, consulte Expressions.

Ejemplo: suma

La siguiente regla de ejemplo comprueba si la suma de todos los valores de una columna supera un umbral dado.

```
Sum "transaction_total" > 500000
Sum "Salary" < 55600 where "Customer_ID < 10"
```

Muestra de reglas dinámicas

- `Sum "ColA" > avg(last(10))`
- `Sum "colA" between min(last(10)) - 1 and max(last(10)) + 1`

Comportamiento nulo

La regla `Sum` ignorará las filas con valores `NULL` al momento de calcular la suma. Por ejemplo:

```
+---+-----------+
|id |units      |
+---+-----------+
|100|0          |
|101|null       |
|102|20         |
```

```
|103|null       |
|104|40         |
+---+-----------+
```

La suma de la columna `units` no considerará las filas 101 y 103 y su resultado será (0 + 20 + 40) = 60.

SchemaMatch

Comprueba si el esquema del conjunto de datos principal coincide con el esquema de un conjunto de datos de referencia. La verificación del esquema se realiza columna por columna. El esquema de dos columnas coincide si los nombres son idénticos y los tipos son idénticos. El orden de las columnas del archivo no importa.

Sintaxis

```
SchemaMatch <REFERENCE_DATASET_ALIAS> <EXPRESSION>
```

- REFERENCE_DATASET_ALIAS: el alias del conjunto de datos de referencia con el que se comparan los esquemas.

  Tipos de columnas compatibles: byte, decimal, doble, flotante, entero, largo, corto
- EXPRESSION: una expresión que se ejecuta en función de la respuesta del tipo de regla para producir un valor booleano. Para obtener más información, consulte Expressions.

Ejemplo: SchemaMatch

La siguiente regla de ejemplo comprueba si el esquema del conjunto de datos principal coincide exactamente con el esquema de un conjunto de datos de referencia.

```
SchemaMatch "reference" = 1.0
```

Singularidad

Comprueba el porcentaje de valores únicos de una columna con respecto a una expresión determinada. Los valores únicos aparecen exactamente una vez.

Sintaxis

```
Uniqueness <COL_NAME> <EXPRESSION>
```

- COL_NAME: el nombre de la columna con la que quiere evaluar la regla de la calidad de los datos.

  Tipos de columnas compatibles: cualquier tipo de columna
- EXPRESSION: una expresión que se ejecuta en función de la respuesta del tipo de regla para producir un valor booleano. Para obtener más información, consulte Expressions.

Ejemplo: porcentaje de singularidad

La siguiente regla de ejemplo comprueba si el porcentaje de valores únicos de una columna coincide con ciertos criterios numéricos.

```
Uniqueness "email" = 1.0
Uniqueness "Customer_ID" != 1.0 where "Customer_ID < 10"
```

Muestra de reglas dinámicas

- `Uniqueness "colA" between min(last(10)) and max(last(10))`
- `Uniqueness "colA" >= avg(last(10))`

UniqueValueRatio

Comprueba la relación de valores únicos de una columna con respecto a una expresión determinada. Una relación de valores únicos es la fracción de valores únicos dividida entre el número de todos los valores distintos de una columna. Los valores únicos aparecen exactamente una vez, mientras que los valores distintos aparecen al menos una vez.

Por ejemplo, el conjunto `[a, a, b]` contiene un valor único (b) y dos valores distintos (a y b). Por lo tanto, la relación de valores únicos del conjunto es ½ = 0,5.

Sintaxis

```
UniqueValueRatio <COL_NAME> <EXPRESSION>
```

- COL_NAME: el nombre de la columna con la que quiere evaluar la regla de la calidad de los datos.

Tipos de columnas compatibles: cualquier tipo de columna

- EXPRESSION: una expresión que se ejecuta en función de la respuesta del tipo de regla para producir un valor booleano. Para obtener más información, consulte Expressions.

Ejemplo: relación de valores únicos

En este ejemplo, se comprueba la relación de valores única de una columna frente a un rango de valores.

```
UniqueValueRatio "test_score" between 0 and 0.5
UniqueValueRatio "Customer_ID" between 0 and 0.9 where "Customer_ID < 10"
```

Muestra de reglas dinámicas

- `UniqueValueRatio "colA" > avg(last(10))`
- `UniqueValueRatio "colA" <= index(last(10),2) + std(last(5))`

DetectAnomalies

Detecta anomalías en una regla de calidad de datos determinada. Cada ejecución de una DetectAnomalies regla tiene como resultado que se guarde el valor evaluado de la regla dada. Cuando se recopilan suficientes datos, el algoritmo de detección de anomalías toma todos los datos históricos de esa regla determinada y ejecuta la detección de anomalías. DetectAnomalies la regla falla cuando se detecta una anomalía. Puede obtener más información sobre qué anomalía se detectó en Observaciones.

Sintaxis

```
    DetectAnomalies <RULE_NAME> <RULE_PARAMETERS>
```

`RULE_NAME`: nombre de la regla que quiere evaluar y detectar anomalías. Reglas admitidas:

- "RowCount"
- “Completeness”
- “Uniqueness”

- “Mean”
- “Sum”
- "StandardDeviation"
- “Entropy”
- "DistinctValuesCount"
- "UniqueValueRatio"
- "ColumnLength"
- "ColumnValues"
- "ColumnCorrelation"

`RULE_PARAMETERS`: algunas reglas requieren parámetros adicionales para poder ejecutarse. Consulte la documentación de reglas dada para ver los parámetros necesarios.

Ejemplo: anomalías para RowCount

Por ejemplo, si queremos detectar RowCount anomalías, proporcionamos un nombre de RowCount regla.

```
DetectAnomalies "RowCount"
```

Ejemplo: Anomalías para ColumnLength

Por ejemplo, si queremos detectar ColumnLength anomalías, proporcionamos ColumnLength como nombre de regla y nombre de columna.

```
DetectAnomalies "ColumnLength" "id"
```

Uso de las API para medir y gestionar la calidad de los datos

En este tema, se describe cómo utilizar las API para medir y gestionar la calidad de los datos.

Contenido

- Requisitos previos
- Cómo trabajar con las recomendaciones de Calidad de datos de AWS Glue
- Trabajar con los conjuntos de reglas de Calidad de datos de AWS Glue
- Cómo trabajar con ejecuciones de Calidad de datos de AWS Glue

- Cómo trabajar con los resultados de Calidad de datos de AWS Glue

Requisitos previos

- Asegúrese de que su versión de boto3 esté actualizada para que incluya la última API de Calidad de datos de AWS Glue.
- Asegúrese de que la versión de AWS CLI esté actualizada para incluir la CLI más reciente.

Si utiliza un trabajo de AWS Glue para ejecutar estas API, puede usar la siguiente opción para actualizar la biblioteca boto3 a la última versión:

```
—additional-python-modules boto3==<version>
```

Cómo trabajar con las recomendaciones de Calidad de datos de AWS Glue

Para iniciar una recomendación de Calidad de datos de AWS Glue, ejecute lo siguiente:

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""
    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client

    def start_data_quality_rule_recommendation_run(self, database_name, table_name,
 role_arn):
        """
        Starts a recommendation run that is used to generate rules when you don't
 know what rules to write. AWS Glue Data Quality analyzes the data and comes up with
 recommendations for a potential ruleset. You can then triage the ruleset and modify
 the generated ruleset to your liking.

        :param database_name: The name of the AWS Glue database which contains the
 dataset.
        :param table_name: The name of the AWS Glue table against which we want a
 recommendation
        :param role_arn: The Amazon Resource Name (ARN) of an AWS Identity and Access
 Management (IAM) role that grants permission to let AWS Glue access the resources it
 needs.
```

```
        """
        try:
            response = self.client.start_data_quality_rule_recommendation_run(
                DataSource={
                    'GlueTable': {
                        'DatabaseName': database_name,
                        'TableName': table_name
                    }
                },
                Role=role_arn
            )
        except ClientError as err:
            logger.error(
                "Couldn't start data quality recommendation run %s. Here's why: %s:
 %s", name,
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return response['RunId']
```

Para una ejecución de recomendaciones, puede utilizar sus recomendaciones `pushDownPredicates` o `catalogPartitionPredicates` para mejorar el rendimiento y ejecutar las recomendaciones solo en particiones específicas de los orígenes de su catálogo.

```
client.start_data_quality_rule_recommendation_run(
            DataSource={
                'GlueTable': {
                    'DatabaseName': database_name,
                    'TableName': table_name,
                    'AdditionalOptions': {
                        'pushDownPredicate': "year=2022"
                    }
                }
            },
            Role=role_arn,
            NumberOfWorkers=2,
            CreatedRulesetName='<rule_set_name>'
  )
```

Para obtener los resultados de una recomendación de Calidad de datos de AWS Glue, ejecute:

```
class GlueWrapper:
```

```
    """Encapsulates AWS Glue actions."""
    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 AWS Glue client.
        """
        self.glue_client = glue_client

    def get_data_quality_rule_recommendation_run(self, run_id):
        """
        Gets the specified recommendation run that was used to generate rules.

        :param run_id: The id of the data quality recommendation run

        """
        try:
            response =
 self.client.get_data_quality_rule_recommendation_run(RunId=run_id)
        except ClientError as err:
            logger.error(
                "Couldn't get data quality recommendation run %. Here's why: %s: %s",
 run_id,
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return response
```

Del objeto de respuesta anterior, puede extraer el conjunto de reglas recomendado por la ejecución para usarlo en los siguientes pasos:

```
print(response['RecommendedRuleset'])

Rules = [
    RowCount between 2000 and 8000,
    IsComplete "col1",
    IsComplete "col2",
    StandardDeviation "col3" between 58138330.8 and 64258155.09,
    ColumnValues "col4" between 1000042965 and 1214474826,
    IsComplete "col5"
]
```

Para obtener una lista de todas las ejecuciones de recomendación que se puedan filtrar y enumerar:

```
response = client.list_data_quality_rule_recommendation_runs(
```

```
    Filter={
        'DataSource': {
            'GlueTable': {
                'DatabaseName': '<database_name>',
                'TableName': '<table_name>'
            }
        }
)
```

Para cancelar las tareas de recomendación de Calidad de datos de AWS Glue existentes:

```
response = client.cancel_data_quality_rule_recommendation_run(
    RunId='dqrun-d4b6b01957fdd79e59866365bf9cb0e40fxxxxxxx'
)
```

Trabajar con los conjuntos de reglas de Calidad de datos de AWS Glue

Para crear un conjunto de reglas de Calidad de datos de AWS Glue:

```
response = client.create_data_quality_ruleset(
    Name='<ruleset_name>',
    Ruleset='Rules = [IsComplete "col1", IsPrimaryKey "col2", RowCount between 2000 and
 8000]',
    TargetTable={
        'TableName': '<table_name>',
        'DatabaseName': '<database_name>'
    }
)
```

Para obtener un conjunto de reglas de calidad de datos:

```
response = client.get_data_quality_ruleset(
    Name='<ruleset_name>'
)
print(response)
```

Puede utilizar esta API para, a continuación, extraer el conjunto de reglas:

```
print(response['Ruleset'])
```

Para enumerar todos los conjuntos de reglas de calidad de datos de una tabla:

```
response = client.list_data_quality_rulesets()
```

Puedes usar la condición de filtro de la API para filtrar todos los conjuntos de reglas adjuntos a una base de datos o tabla específica:

```
response = client.list_data_quality_rulesets(
    Filter={
        'TargetTable': {
            'TableName': '<table_name>',
            'DatabaseName': '<database_name>'
        }
    },
)
```

Para actualizar un conjunto de reglas de calidad de datos:

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""
    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 AWS Glue client.
        """
        self.glue_client = glue_client

    def update_data_quality_ruleset(self, ruleset_name, ruleset_string):
        """
        Update an AWS Glue Data Quality Ruleset

        :param ruleset_name: The name of the AWS Glue Data Quality ruleset to update
        :param ruleset_string: The DQDL ruleset string to update the ruleset with

        """
        try:
            response = self.client.update_data_quality_ruleset(
                Name=ruleset_name,
                Ruleset=ruleset_string
            )
        except ClientError as err:
            logger.error(
                "Couldn't update the AWS Glue Data Quality ruleset. Here's why: %s:
 %s",
                err.response['Error']['Code'], err.response['Error']['Message'])
```

```
            raise
        else:
            return response
```

Para eliminar un conjunto de reglas de la calidad de los datos:

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""
    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 AWS Glue client.
        """
        self.glue_client = glue_client

    def delete_data_quality_ruleset(self, ruleset_name):
        """
        Delete a AWS Glue Data Quality Ruleset

        :param ruleset_name: The name of the AWS Glue Data Quality ruleset to delete

        """
        try:
            response = self.client.delete_data_quality_ruleset(
                Name=ruleset_name
            )
        except ClientError as err:
            logger.error(
                "Couldn't delete the AWS Glue Data Quality ruleset. Here's why: %s:
 %s",
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return response
```

Cómo trabajar con ejecuciones de Calidad de datos de AWS Glue

Para iniciar una ejecución de Calidad de datos de AWS Glue:

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""
    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 AWS Glue client.
```

```
        """
        self.glue_client = glue_client

    def start_data_quality_ruleset_evaluation_run(self, database_name, table_name,
role_name, ruleset_list):
        """
        Start an AWS Glue Data Quality evaluation run

        :param database_name: The name of the AWS Glue database which contains the
dataset.
        :param table_name: The name of the AWS Glue table against which we want to
evaluate.
        :param role_arn: The Amazon Resource Name (ARN) of an AWS Identity and Access
Management (IAM) role that grants permission to let AWS Glue access the resources it
needs.
        :param ruleset_list: The list of AWS Glue Data Quality ruleset names to
evaluate.

        """
        try:
            response = client.start_data_quality_ruleset_evaluation_run(
                DataSource={
                    'GlueTable': {
                        'DatabaseName': database_name,
                        'TableName': table_name
                    }
                },
                Role=role_name,
                RulesetNames=ruleset_list
            )
        except ClientError as err:
            logger.error(
                "Couldn't start the AWS Glue Data Quality Run. Here's why: %s: %s",
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return response['RunId']
```

Recuerde que puede pasar un parámetro `pushDownPredicate` o `catalogPartitionPredicate` para garantizar que la ejecución de calidad de los datos solo se dirija a un conjunto específico de particiones de la tabla de catálogo. Por ejemplo:

```
response = client.start_data_quality_ruleset_evaluation_run(
```

```
    DataSource={
        'GlueTable': {
            'DatabaseName': '<database_name>',
            'TableName': '<table_name>',
            'AdditionalOptions': {
                'pushDownPredicate': 'year=2023'
            }
        }
    },
    Role='<role_name>',
    NumberOfWorkers=5,
    Timeout=123,
    AdditionalRunOptions={
        'CloudWatchMetricsEnabled': False
    },
    RulesetNames=[
        '<ruleset_name>',
    ]
)
```

Para obtener información sobre una ejecución de Calidad de datos de AWS Glue:

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""
    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 AWS Glue client.
        """
        self.glue_client = glue_client

    def get_data_quality_ruleset_evaluation_run(self, run_id):
        """
        Get details about an AWS Glue Data Quality Run

        :param run_id: The AWS Glue Data Quality run ID to look up

        """
        try:
            response = self.client.get_data_quality_ruleset_evaluation_run(
                RunId=run_id
            )
        except ClientError as err:
            logger.error(
```

```
                "Couldn't look up the AWS Glue Data Quality run ID. Here's why: %s:
 %s",
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return response
```

Para obtener los resultados de una ejecución de Calidad de datos de AWS Glue:

Para una ejecución de Calidad de datos de AWS Glue determinada, puede extraer los resultados de la evaluación de la ejecución mediante el siguiente método:

```
response = client.get_data_quality_ruleset_evaluation_run(
    RunId='d4b6b01957fdd79e59866365bf9cb0e40fxxxxxxx'
)

resultID = response['ResultIds'][0]

response = client.get_data_quality_result(
    ResultId=resultID
)

print(response['RuleResults'])
```

Para ver una lista de todas sus ejecuciones de Calidad de datos de AWS Glue:

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""
    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 AWS Glue client.
        """
        self.glue_client = glue_client

    def list_data_quality_ruleset_evaluation_runs(self, database_name, table_name):
        """
        Lists all the AWS Glue Data Quality runs against a given table

        :param database_name: The name of the database where the data quality runs
        :param table_name: The name of the table against which the data quality runs
 were created

        """
```

```
        try:
            response = self.client.list_data_quality_ruleset_evaluation_runs(
                Filter={
                    'DataSource': {
                        'GlueTable': {
                            'DatabaseName': database_name,
                            'TableName': table_name
                        }
                    }
                }
            )
        except ClientError as err:
            logger.error(
                "Couldn't list the AWS Glue Quality runs. Here's why: %s: %s",
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return response
```

Puede modificar la cláusula de filtro para que solo muestre los resultados entre tiempos específicos o si se ejecutan en tablas específicas.

Para detener una ejecución continua de Calidad de datos de AWS Glue:

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""
    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 AWS Glue client.
        """
        self.glue_client = glue_client

    def cancel_data_quality_ruleset_evaluation_run(self, result_id):
        """
        Cancels a given AWS Glue Data Quality run

        :param result_id: The result id of a AWS Glue Data Quality run to cancel

        """
        try:
            response = self.client.cancel_data_quality_ruleset_evaluation_run(
                ResultId=result_id
            )
```

```
        except ClientError as err:
            logger.error(
                "Couldn't cancel the AWS Glue Data Quality run. Here's why: %s: %s",
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return response
```

Cómo trabajar con los resultados de Calidad de datos de AWS Glue

Para obtener los resultados de ejecución de Calidad de datos de AWS Glue:

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""
    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 AWS Glue client.
        """
        self.glue_client = glue_client

    def get_data_quality_result(self, result_id):
        """
        Outputs the result of an AWS Glue Data Quality Result

        :param result_id: The result id of an AWS Glue Data Quality run

        """
        try:
            response = self.client.get_data_quality_result(
                ResultId=result_id
            )
        except ClientError as err:
            logger.error(
                "Couldn't get the AWS Glue Data Quality result. Here's why: %s: %s",
                err.response['Error']['Code'], err.response['Error']['Message'])
            raise
        else:
            return response
```

Para cancelar las tareas de recomendación de Calidad de datos de AWS Glue existentes:

Con un identificador de ejecución de Calidad de datos de AWS Glue, puede extraer el identificador del resultado para obtener los resultados reales, tal y como se muestra a continuación:

```
response = client.get_data_quality_ruleset_evaluation_run(
    RunId='dqrun-abca77ee126abe1378c1da1ae0750xxxxxxxx'
)

resultID = response['ResultIds'][0]

response = client.get_data_quality_result(
    ResultId=resultID
)

print(resp['RuleResults'])
```

Configuración de alertas, implementaciones y programación

En este tema se describe cómo configurar alertas, despliegues y programación para AWS Glue Data Quality.

Contenido

- Configuración de alertas y notificaciones en la EventBridge integración de Amazon
  - Opciones de configuración adicionales para el patrón de eventos
  - Formatación de notificaciones como correos electrónicos
- Configure alertas y notificaciones en la integración CloudWatch
- Cómo consultar los resultados de calidad de los datos para crear paneles
- Implementación de reglas de calidad de datos mediante AWS CloudFormation
- Programación de reglas de calidad de datos

Configuración de alertas y notificaciones en la EventBridge integración de Amazon

AWS Glue Data Quality admite la publicación de EventBridge eventos, que se emiten al completar una evaluación del conjunto de reglas de calidad de datos. De este modo, puede configurar fácilmente alertas cuando fallen las reglas de calidad de datos.

Este es un ejemplo de un evento al evaluar los conjuntos de reglas de calidad de los datos en el Data Catalog. Con esta información, puedes revisar los datos que están disponibles en Amazon EventBridge. Puede realizar llamadas a la API adicionales para obtener más información. Por

ejemplo, llama a la API `get_data_quality_result` con el ID de resultado para obtener los detalles de una ejecución concreta.

```
{
    "version":"0",
    "id":"abcdef00-1234-5678-9abc-def012345678",
    "detail-type":"Data Quality Evaluation Results Available",
    "source":"aws.glue-dataquality",
    "account":"123456789012",
    "time":"2017-09-07T18:57:21Z",
    "region":"us-west-2",
    "resources":[],
    "detail":{
        "context": {
                    "contextType": "GLUE_DATA_CATALOG",
                    "runId":"dqrun-12334567890",
                    "databaseName": "db-123",
                    "tableName": "table-123",
                    "catalogId": "123456789012"
                    },
        "resultID": "dqresult-12334567890",
        "rulesetNames": ["rulset1"],
        "state":"SUCCEEDED",
        "score": 1.00,
        "rulesSucceeded": 100,
        "rulesFailed": 0,
        "rulesSkipped": 0
    }
}
```

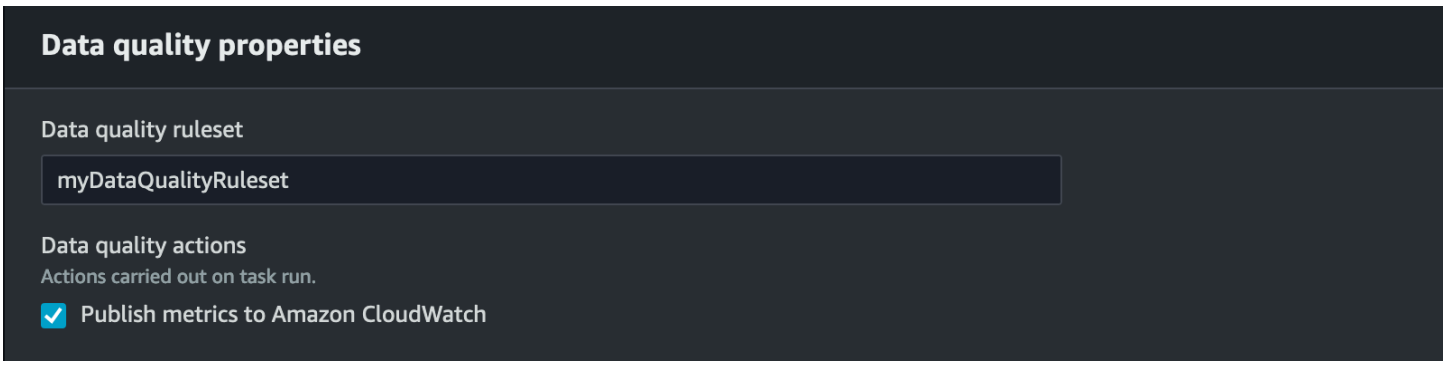
Este es un ejemplo de evento que se publica al evaluar los conjuntos de reglas de calidad de los datos en las libretas AWS Glue ETL o AWS Glue Studio.

```
{
    "version":"0",
    "id":"abcdef00-1234-5678-9abc-def012345678",
    "detail-type":"Data Quality Evaluation Results Available",
    "source":"aws.glue-dataquality",
    "account":"123456789012",
    "time":"2017-09-07T18:57:21Z",
    "region":"us-west-2",
    "resources":[],
    "detail":{
```

```
        "context": {
                    "contextType": "GLUE_JOB",
                    "jobId": "jr-12334567890",
                    "jobName": "dq-eval-job-1234",
                    "evaluationContext": "",
                    }
        "resultID": "dqresult-12334567890",
        "rulesetNames": ["rulset1"],
        "state":"SUCCEEDED",
        "score": 1.00
        "rulesSucceeded": 100,
        "rulesFailed": 0,
        "rulesSkipped": 0
    }
}
```

Para que la evaluación de la calidad de los datos se ejecute tanto en el catálogo de datos como en los trabajos de ETL, la Amazon CloudWatch opción Publicar métricas en, que está seleccionada de forma predeterminada, debe permanecer seleccionada para que la EventBridge publicación funcione.

Configurar las EventBridge notificaciones



Para recibir los eventos emitidos y definir los objetivos, debes configurar EventBridge las reglas de Amazon. Para crear reglas:

1. Abre la EventBridge consola de Amazon.
2. Elija Reglas en la sección Buses de la barra de navegación.
3. Elija Create Rule.
4. En Definir los detalles de las reglas:
   a. En Nombre, escriba `myDQRule`.
   b. Ingrese la descripción (opcional).

c. Para el bus de eventos, seleccione su bus de eventos. Si no tiene uno, déjelo como predeterminado.

d. En Tipo de regla, seleccione Regla con un patrón de evento y luego elija Siguiente.

5. En Construir patrón de evento:

a. Para la fuente del evento, selecciona AWS eventos o eventos EventBridge asociados.

b. Omita la sección de eventos de muestra.

c. Para el método de creación, seleccione Usar forma de patrón.

d. Para el patrón de eventos:

i. Seleccione Servicios de AWS para el origen del evento.

ii. Seleccione Glue Data Quality para obtener AWS servicio.

iii. Seleccione Resultados de la evaluación de la calidad de los datos disponibles para el tipo de evento.

iv. Seleccione FALLA para estado(s) específico(s). A continuación, puede ver un patrón de eventos similar al siguiente:

```
{
  "source": ["aws.glue-dataquality"],
  "detail-type": ["Data Quality Evaluation Results Available"],
  "detail": {
    "state": ["FAILED"]
  }
}
```

v. Para más opciones de configuración, consulte Opciones de configuración adicionales para el patrón de eventos.

6. En Objetivo(s) seleccionado(s):

a. Para Tipos de objetivos, seleccione el Servicio de AWS .

b. Utilice el menú desplegable Seleccione un destino para elegir el AWS servicio al que desee conectarse (SNS, Lambda, SQS, etc.) y, a continuación, seleccione Siguiente.

7. En Configurar etiqueta(s), haga clic en Agregar nuevas etiquetas para agregar etiquetas opcionales y, a continuación, seleccione Siguiente.

8. Aparecerá una página de resumen de todas las selecciones. Selecciona Crear regla en la parte inferior.

Opciones de configuración adicionales para el patrón de eventos

Además de filtrar el evento en función del éxito o el fracaso, puede que desee filtrar más los eventos según distintos parámetros.

Para ello, vaya a la sección Patrón de eventos y seleccione Editar patrón para especificar parámetros adicionales. Tenga en cuenta que los campos del patrón de eventos distinguen entre mayúsculas y minúsculas. A continuación, se muestran ejemplos de cómo configurar el patrón de eventos.

Para capturar eventos de una tabla en particular que evalúe conjuntos de reglas específicos, utilice este tipo de patrón:

```
{
  "source": ["aws.glue-dataquality"],
  "detail-type": ["Data Quality Evaluation Results Available"],
  "detail": {
    "context": {
      "contextType": ["GLUE_DATA_CATALOG"],
      "databaseName": "db-123",
       "tableName": "table-123",
    },
    "rulesetNames": ["ruleset1", "ruleset2"]
    "state": ["FAILED"]
  }
}
```

Para capturar eventos de trabajos específicos de la experiencia ETL, utilice este tipo de patrón:

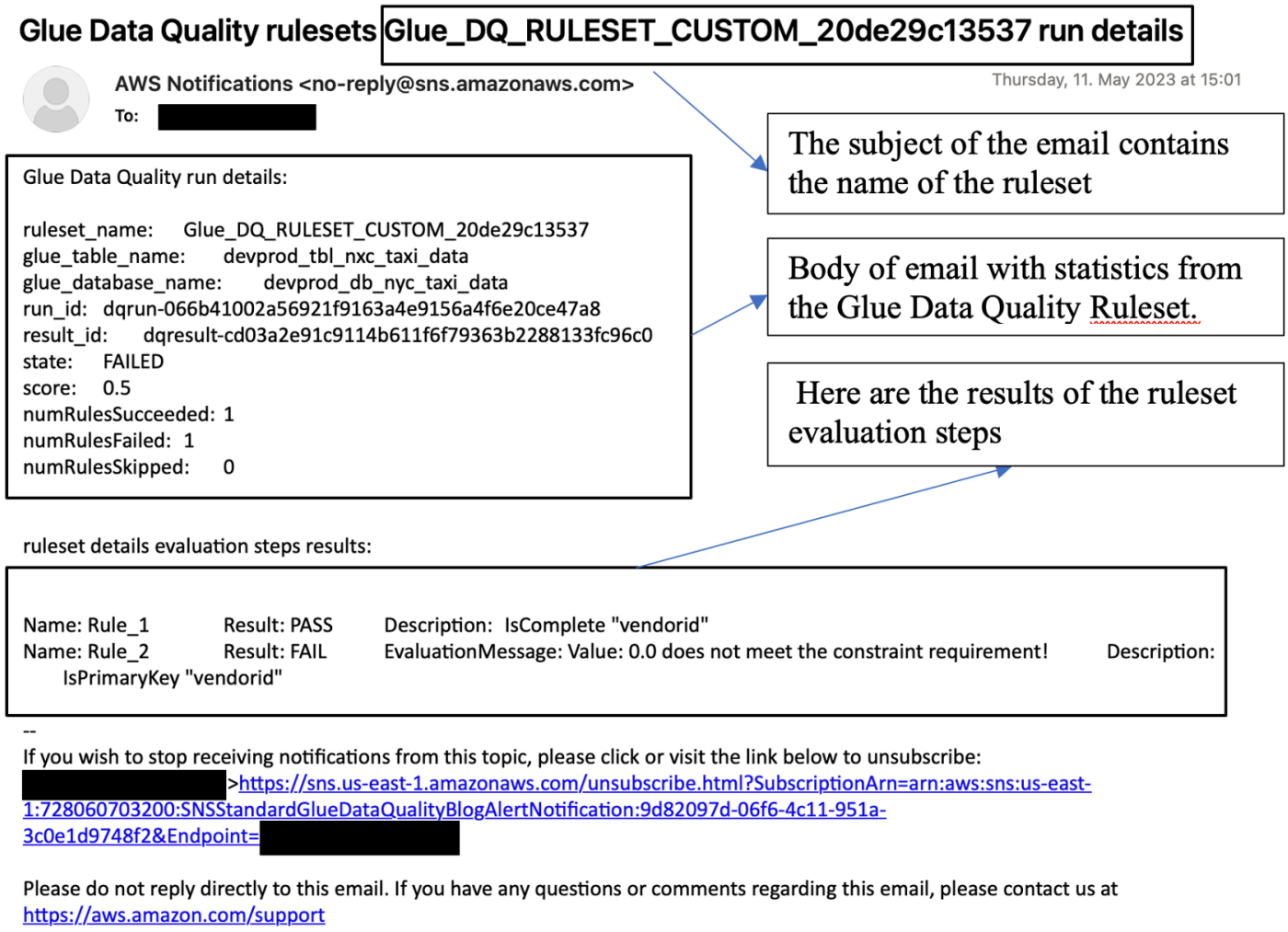
```
{
  "source": ["aws.glue-dataquality"],
  "detail-type": ["Data Quality Evaluation Results Available"],
  "detail": {
    "context": {
      "contextType": ["GLUE_JOB"],
      "jobName": ["dq_evaluation_job1", "dq_evaluation_job2"]
    },
    "state": ["FAILED"]
  }
}
```

Para capturar eventos con una puntuación inferior a un umbral específico (por ejemplo, 70 %):

```
{
  "source": ["aws.glue-dataquality"],
  "detail-type": ["Data Quality Evaluation Results Available"],
  "detail": {
    "score": [{
      "numeric": ["<=", 0.7]
    }]
  }
}
```

Formatación de notificaciones como correos electrónicos

A veces es necesario enviar una notificación por correo electrónico con un formato adecuado a los equipos de la empresa. Para ello, puede utilizar Amazon EventBridge y AWS Lambda.



El siguiente código de ejemplo se puede utilizar para formatear las notificaciones de calidad de los datos a fin de generar correos electrónicos.

```
import boto3
import json
from datetime import datetime

sns_client = boto3.client('sns')
glue_client = boto3.client('glue')

sns_topic_arn = 'arn:aws:sns:<region-code>:<account-id>:<sns-topic-name>'



def lambda_handler(event, context):
    log_metadata = {}
    message_text = ""
    subject_text = ""

    if event['detail']['context']['contextType'] == 'GLUE_DATA_CATALOG':
        log_metadata['ruleset_name'] = str(event['detail']['rulesetNames'][0])
        log_metadata['tableName'] = str(event['detail']['context']['tableName'])
        log_metadata['databaseName'] = str(event['detail']['context']['databaseName'])
        log_metadata['runId'] = str(event['detail']['context']['runId'])
        log_metadata['resultId'] = str(event['detail']['resultId'])
        log_metadata['state'] = str(event['detail']['state'])
        log_metadata['score'] = str(event['detail']['score'])
        log_metadata['numRulesSucceeded'] = str(event['detail']['numRulesSucceeded'])
        log_metadata['numRulesFailed'] = str(event['detail']['numRulesFailed'])
        log_metadata['numRulesSkipped'] = str(event['detail']['numRulesSkipped'])

        message_text += "Glue Data Quality run details:\n"
        message_text += "ruleset_name: {}\n".format(log_metadata['ruleset_name'])
        message_text += "glue_table_name: {}\n".format(log_metadata['tableName'])
        message_text += "glue_database_name: {}\n".format(log_metadata['databaseName'])
        message_text += "run_id: {}\n".format(log_metadata['runId'])
        message_text += "result_id: {}\n".format(log_metadata['resultId'])
        message_text += "state: {}\n".format(log_metadata['state'])
        message_text += "score: {}\n".format(log_metadata['score'])
        message_text += "numRulesSucceeded:
 {}\n".format(log_metadata['numRulesSucceeded'])
        message_text += "numRulesFailed: {}\n".format(log_metadata['numRulesFailed'])
        message_text += "numRulesSkipped: {}\n".format(log_metadata['numRulesSkipped'])
```

```
        subject_text = "Glue Data Quality ruleset {} run
details".format(log_metadata['ruleset_name'])

    else:
        log_metadata['ruleset_name'] = str(event['detail']['rulesetNames'][0])
        log_metadata['jobName'] = str(event['detail']['context']['jobName'])
        log_metadata['jobId'] = str(event['detail']['context']['jobId'])
        log_metadata['resultId'] = str(event['detail']['resultId'])
        log_metadata['state'] = str(event['detail']['state'])
        log_metadata['score'] = str(event['detail']['score'])

        log_metadata['numRulesSucceeded'] = str(event['detail']['numRulesSucceeded'])
        log_metadata['numRulesFailed'] = str(event['detail']['numRulesFailed'])
        log_metadata['numRulesSkipped'] = str(event['detail']['numRulesSkipped'])

        message_text += "Glue Data Quality run details:\n"
        message_text += "ruleset_name: {}\n".format(log_metadata['ruleset_name'])
        message_text += "glue_job_name: {}\n".format(log_metadata['jobName'])
        message_text += "job_id: {}\n".format(log_metadata['jobId'])
        message_text += "result_id: {}\n".format(log_metadata['resultId'])
        message_text += "state: {}\n".format(log_metadata['state'])
        message_text += "score: {}\n".format(log_metadata['score'])
        message_text += "numRulesSucceeded:
{}\n".format(log_metadata['numRulesSucceeded'])
        message_text += "numRulesFailed: {}\n".format(log_metadata['numRulesFailed'])
        message_text += "numRulesSkipped: {}\n".format(log_metadata['numRulesSkipped'])

        subject_text = "Glue Data Quality ruleset {} run
details".format(log_metadata['ruleset_name'])

    resultID = str(event['detail']['resultId'])
    response = glue_client.get_data_quality_result(ResultId=resultID)
    RuleResults = response['RuleResults']
    message_text += "\n\nruleset details evaluation steps results:\n\n"
    subresult_info = []

    for dic in RuleResults:
        subresult = "Name: {}\t\tResult: {}\t\tDescription: \t{}".format(dic['Name'],
dic['Result'], dic['Description'])
        if 'EvaluationMessage' in dic:
            subresult += "\t\tEvaluationMessage: {}".format(dic['EvaluationMessage'])
        subresult_info.append({
            'Name': dic['Name'],
```

```
            'Result': dic['Result'],
            'Description': dic['Description'],
            'EvaluationMessage': dic.get('EvaluationMessage', '')
        })
        message_text += "\n" + subresult

    log_metadata['resultrun'] = subresult_info



    sns_client.publish(
        TopicArn=sns_topic_arn,
        Message=message_text,
        Subject=subject_text
    )

    return {
        'statusCode': 200,
        'body': json.dumps('Message published to SNS topic')
    }
```

Configure alertas y notificaciones en la integración CloudWatch

Nuestro enfoque recomendado es configurar alertas de calidad de datos con Amazon EventBridge, ya que Amazon EventBridge requiere una configuración única para alertar a los clientes. Sin embargo, algunos clientes prefieren Amazon CloudWatch debido a su familiaridad. Para estos clientes, ofrecemos la integración con Amazon CloudWatch.

Cada evaluación de calidad de datos de AWS Glue emite un par de métricas denominadas `glue.data.quality.rules.passed` (que indican el número de reglas que se han aprobado) y `glue.data.quality.rules.failed` (que indican el número de reglas no aprobadas) por ejecución de calidad de datos. Puede usar esta métrica emitida para crear alarmas que alerten a los usuarios si una determinada cantidad de calidad de datos cae por debajo de un umbral. Para empezar a configurar una alarma que envíe un correo electrónico a través de una notificación de Amazon SNS, siga los pasos que se indican a continuación:

Para empezar a configurar una alarma que envíe un correo electrónico a través de una notificación de Amazon SNS, siga los pasos que se indican a continuación:

1. Abre la CloudWatch consola de Amazon.

2. Seleccione Todas las métricas en Métricas. Verá un espacio de nombres adicional en Espacios de nombres personalizados denominado Calidad de datos de Glue.

> **Note**
>
> Al iniciar una ejecución de AWS Glue Data Quality, asegúrate de que la CloudWatch casilla Publicar métricas en Amazon esté activada. De lo contrario, las métricas de esa ejecución concreta no se publicarán en Amazon CloudWatch.

   En el espacio de nombres `Glue Data Quality`, puede ver las métricas que se emiten por tabla y por conjunto de reglas. Para este tema, utilizaremos la `glue.data.quality.rules.failed` regla y la alarma si este valor supera 1 (lo que indica que, si vemos un número de evaluaciones de reglas fallidas superior a 1, queremos que se nos notifique).
3. Para crear la alarma, seleccione Todas las alarmas en Alarmas.
4. Elija Create alarm (Crear alarma).
5. Elija Select Metric (Seleccionar métrica).
6. Seleccione la métrica `glue.data.quality.rules.failed` correspondiente a la tabla que creó y, a continuación, elija Seleccionar métrica.
7. En la pestaña Especificar métrica y condiciones, en la sección Métricas:
   a. En Statistic (Estadística), elija Sum (Suma).
   b. Para Periodo, seleccione 1 minuto.
8. En la sección Condiciones:
   a. En Threshold type (Tipo de umbral), elija Static (Estático).
   b. Para Siempre que glue.data.quality.rules.failed sea…, seleccione Mayor que/Igual a.
   c. Para que..., ingrese 1 como valor umbral.

   Estas selecciones implican que si la métrica `glue.data.quality.rules.failed` emite un valor mayor o igual a 1, activaremos una alarma. Sin embargo, si no hay datos, los consideraremos aceptables.
9. Elija Siguiente.
10. En Acciones de configuración:
    a. Para la sección Desencadenador de estado de alarma, elija En alarma.

b. Para la sección Enviar una notificación al siguiente tema de SNS, seleccione Crear un tema nuevo para enviar una notificación a través de un nuevo tema de SNS.

c. En Puntos de conexión de correo electrónico que recibirán la notificación, ingrese una dirección de correo electrónico. Luego haga clic en Crear tema.

d. Elija Siguiente.

11 En Nombre de la alarma, ingrese `myFirstDQAlarm` y, a continuación, seleccione Siguiente.

12 Aparecerá una página de resumen de todas las selecciones. Seleccione Crear alarma en la parte inferior.

Ahora puedes ver la alarma que se está creando desde el panel de CloudWatch alarmas de Amazon.

Cómo consultar los resultados de calidad de los datos para crear paneles

Es posible que desee crear un panel para mostrar los resultados de calidad de sus datos. Hay dos formas de hacer esto:

Configure Amazon EventBridge con el siguiente código para escribir los datos en Amazon S3:

```
import boto3
import json
from datetime import datetime


s3_client = boto3.client('s3')
glue_client = boto3.client('glue')


s3_bucket = 's3-bucket-name'

def write_logs(log_metadata):
    try:
        filename = datetime.now().strftime("%m%d%Y%H%M%S") + ".json"
        key_opts = {
            'year': datetime.now().year,
            'month': "{:02d}".format(datetime.now().month),
            'day': "{:02d}".format(datetime.now().day),
            'filename': filename
        }
        s3key = "gluedataqualitylogs/year={year}/month={month}/day={day}/
{filename}".format(**key_opts)
```

```
        s3_client.put_object(Bucket=s3_bucket, Key=s3key,
 Body=json.dumps(log_metadata))
    except Exception as e:
        print(f'Error writing logs to S3: {e}')


def lambda_handler(event, context):
    log_metadata = {}
    message_text = ""
    subject_text = ""

    if event['detail']['context']['contextType'] == 'GLUE_DATA_CATALOG':
        log_metadata['ruleset_name'] = str(event['detail']['rulesetNames'][0])
        log_metadata['tableName'] = str(event['detail']['context']['tableName'])
        log_metadata['databaseName'] = str(event['detail']['context']['databaseName'])
        log_metadata['runId'] = str(event['detail']['context']['runId'])
        log_metadata['resultId'] = str(event['detail']['resultId'])
        log_metadata['state'] = str(event['detail']['state'])
        log_metadata['score'] = str(event['detail']['score'])
        log_metadata['numRulesSucceeded'] = str(event['detail']['numRulesSucceeded'])
        log_metadata['numRulesFailed'] = str(event['detail']['numRulesFailed'])
        log_metadata['numRulesSkipped'] = str(event['detail']['numRulesSkipped'])

        message_text += "Glue Data Quality run details:\n"
        message_text += "ruleset_name: {}\n".format(log_metadata['ruleset_name'])
        message_text += "glue_table_name: {}\n".format(log_metadata['tableName'])
        message_text += "glue_database_name: {}\n".format(log_metadata['databaseName'])
        message_text += "run_id: {}\n".format(log_metadata['runId'])
        message_text += "result_id: {}\n".format(log_metadata['resultId'])
        message_text += "state: {}\n".format(log_metadata['state'])
        message_text += "score: {}\n".format(log_metadata['score'])
        message_text += "numRulesSucceeded:
 {}\n".format(log_metadata['numRulesSucceeded'])
        message_text += "numRulesFailed: {}\n".format(log_metadata['numRulesFailed'])
        message_text += "numRulesSkipped: {}\n".format(log_metadata['numRulesSkipped'])

        subject_text = "Glue Data Quality ruleset {} run
 details".format(log_metadata['ruleset_name'])

    else:
        log_metadata['ruleset_name'] = str(event['detail']['rulesetNames'][0])
        log_metadata['jobName'] = str(event['detail']['context']['jobName'])
        log_metadata['jobId'] = str(event['detail']['context']['jobId'])
        log_metadata['resultId'] = str(event['detail']['resultId'])
```

```
        log_metadata['state'] = str(event['detail']['state'])
        log_metadata['score'] = str(event['detail']['score'])

        log_metadata['numRulesSucceeded'] = str(event['detail']['numRulesSucceeded'])
        log_metadata['numRulesFailed'] = str(event['detail']['numRulesFailed'])
        log_metadata['numRulesSkipped'] = str(event['detail']['numRulesSkipped'])

        message_text += "Glue Data Quality run details:\n"
        message_text += "ruleset_name: {}\n".format(log_metadata['ruleset_name'])
        message_text += "glue_job_name: {}\n".format(log_metadata['jobName'])
        message_text += "job_id: {}\n".format(log_metadata['jobId'])
        message_text += "result_id: {}\n".format(log_metadata['resultId'])
        message_text += "state: {}\n".format(log_metadata['state'])
        message_text += "score: {}\n".format(log_metadata['score'])
        message_text += "numRulesSucceeded:
{}\n".format(log_metadata['numRulesSucceeded'])
        message_text += "numRulesFailed: {}\n".format(log_metadata['numRulesFailed'])
        message_text += "numRulesSkipped: {}\n".format(log_metadata['numRulesSkipped'])

        subject_text = "Glue Data Quality ruleset {} run
details".format(log_metadata['ruleset_name'])

    resultID = str(event['detail']['resultId'])
    response = glue_client.get_data_quality_result(ResultId=resultID)
    RuleResults = response['RuleResults']
    message_text += "\n\nruleset details evaluation steps results:\n\n"
    subresult_info = []

    for dic in RuleResults:
        subresult = "Name: {}\t\tResult: {}\t\tDescription: \t{}".format(dic['Name'],
dic['Result'], dic['Description'])
        if 'EvaluationMessage' in dic:
            subresult += "\t\tEvaluationMessage: {}".format(dic['EvaluationMessage'])
        subresult_info.append({
            'Name': dic['Name'],
            'Result': dic['Result'],
            'Description': dic['Description'],
            'EvaluationMessage': dic.get('EvaluationMessage', '')
        })
        message_text += "\n" + subresult

    log_metadata['resultrun'] = subresult_info

    write_logs(log_metadata)
```

```
    return {
        'statusCode': 200,
        'body': json.dumps('Message published to SNS topic')
    }
```

Tras escribir en Amazon S3, puede utilizar los rastreadores de AWS Glue para registrarse en Athena y consultar las tablas.

Configure una ubicación de Amazon S3 durante una evaluación de la calidad de los datos:

Al ejecutar tareas de calidad de datos en el catálogo de datos de AWS AWS Glue o en la ETL de Glue, puede proporcionar una ubicación de Amazon S3 para escribir los resultados de calidad de los datos en Amazon S3. Puede usar la siguiente sintaxis para crear una tabla que haga referencia al objetivo para leer los resultados de calidad de los datos.

Tenga en cuenta que debe ejecutar las consultas `CREATE EXTERNAL TABLE` y `MSCK REPAIR TABLE` por separado.

```
CREATE EXTERNAL TABLE <my_table_name>(
    catalogid string,
    databasename string,
    tablename string,
    dqrunid string,
    evaluationstartedon timestamp,
    evaluationcompletedon timestamp,
    rule string,
    outcome string,
    failurereason string,
    evaluatedmetrics string)
PARTITIONED BY (
    `year` string,
    `month` string,
    `day` string)
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
WITH SERDEPROPERTIES (

 'paths'='catalogId,databaseName,dqRunId,evaluatedMetrics,evaluationCompletedOn,evaluationStart
STORED AS INPUTFORMAT 'org.apache.hadoop.mapred.TextInputFormat'
OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION 's3://glue-s3-dq-bucket-us-east-2-results/'
```

```
TBLPROPERTIES (
    'classification'='json',
    'compressionType'='none',
    'typeOfData'='file');
```

```
MSCK REPAIR TABLE <my_table_name>;
```

Una vez que haya creado la tabla anterior, podrá ejecutar consultas analíticas con Amazon Athena.

Implementación de reglas de calidad de datos mediante AWS CloudFormation

Puede utilizarlas AWS CloudFormation para crear reglas de calidad de datos. Para obtener más información, consulte AWS CloudFormation AWS Glue.

Programación de reglas de calidad de datos

Puede programar las reglas de calidad de los datos mediante los métodos siguientes:

- Programe las reglas de calidad de los datos desde el catálogo de datos: los usuarios de ningún código pueden utilizar esta opción para programar fácilmente sus escaneos de calidad de datos. AWS Glue Data Quality creará el cronograma en Amazon EventBridge. Para programar las reglas de calidad de los datos:
  - Navegue hasta el conjunto de reglas y haga clic en Ejecutar.
  - En la Frecuencia de ejecución, seleccione la programación deseada y proporcione un Nombre de tarea. El nombre de esta tarea es el nombre de su programa EventBridge.
- Utilice Amazon EventBridge y AWS Step Functions para organizar las evaluaciones y recomendaciones de las normas de calidad de los datos.

Solución de errores de calidad de datos de AWS Glue

Si encuentra errores en la calidad de los datos de AWS Glue, utilice las siguientes soluciones para ayudarle a encontrar el origen de los problemas y corregirlos.

Contenido

- Error: falta el módulo AWS Glue Data Quality
- Error: permisos de AWS Lake Formation insuficientes

- Error: los conjuntos de reglas no disponen de un nombre único
- Error: tablas con caracteres especiales
- Error: error de desbordamiento con un conjunto de reglas grande
- Error: el estado general de la regla falló
- AnalysisException: No se ha podido verificar la existencia de la base de datos predeterminada
- Mensaje de error: el mapa de teclas proporcionado no es adecuado para determinados marcos de datos
- Excepción en la clase de usuario: java.lang. RuntimeException : No se pudieron recuperar los datos. Compruebe los inicios de sesión CloudWatch para obtener más detalles
- ERROR DE INICIO: error al descargar desde S3 para bucket
- InvalidInputException (estado: 400): DataQuality las reglas no se pueden analizar
- Error: EventBridge no activa los trabajos de calidad de datos de Glue según la programación que configuré
- Errores de CustomSQL
- Reglas dinámicas
- Excepción en la clase de usuario: org.apache.spark.sql. AnalysisException: org.apache.hadoop.hive.ql.metadata. HiveException
- UNCLASSIFIED_ERROR; IllegalArgumentException: Error de análisis: no se proporcionaron reglas ni analizadores., no hay una alternativa viable en la entrada

Error: falta el módulo AWS Glue Data Quality

Mensaje de error: ningún módulo se llama “awsgluedq”.

Solución: este error se produce al ejecutar AWS Glue Data Quality en una versión no compatible. AWS La calidad de datos de Glue solo se admite en la versión 3.0 y posteriores de Glue.

Error: permisos de AWS Lake Formation insuficientes

Mensaje de error: Excepción en la clase de usuario`com.amazonaws.services.glue.model.AccessDeniedException`: permisos de Lake Formation insuficientes en impact_sdg_involvement (servicio:; código de estado: 400; código de error:AWS Glue; ID de solicitud: AccessDeniedException 465ae693-b7ba-4df0-a4e4-6b17xxxxxxxx; proxy: nulo).

Resolución: Debe proporcionar permisos suficientes en AWS Lake Formation.

Error: los conjuntos de reglas no disponen de un nombre único

Mensaje de error: excepción en la clase de usuario:... services.glue.model. AlreadyExistsException: Ya existe otro conjunto de reglas con el mismo nombre.

Solución: los conjuntos de reglas son globales y es necesario que sean únicos.

Error: tablas con caracteres especiales

Mensaje de error: excepción en la clase de usuario: org.apache.spark.sql. AnalysisException: no se puede resolver «C» dadas las columnas de entrada: [primary.data_end_time, primary.data_start_time, primary.end_time, primary.last_updated, primary.message, primary.process_date, primary.rowhash, primary.run_by, primary.run_id, primary.start_time, primary.status]; línea 1 pos 44;.

Resolución: actualmente existe una limitación que impide que AWS Glue Data Quality se ejecute en tablas con caracteres especiales como «.».

Error: error de desbordamiento con un conjunto de reglas grande

Mensaje de error: excepción en la clase de usuario: java.lang. StackOverflowError.

Solución: si tiene un conjunto de reglas grande de más de 2000 reglas, es posible que se produzca este problema. Divida sus reglas en varios conjuntos de reglas.

Error: el estado general de la regla falló

Condición de error: mi conjunto de reglas es correcto, pero el estado general de mi regla tiene errores.

Solución: lo más probable es que este error se haya producido porque has elegido la opción de publicar las métricas en Amazon CloudWatch mientras publicas. Si tu conjunto de datos está en una VPC, es posible que tu VPC no permita que AWS Glue publique métricas en Amazon. CloudWatch En este caso, >debes configurar un punto de conexión para que tu VPC acceda a Amazon. CloudWatch

AnalysisException: No se ha podido verificar la existencia de la base de datos predeterminada

Condición de error AnalysisException: no se puede comprobar la existencia de la base de datos predeterminada: com.amazonaws.services.glue.model. AccessDeniedException: Permisos de Lake Formation insuficientes por defecto (servicio:AWS Glue; código de estado: 400; código de error:; ID de solicitud: XXXXXXXX-XXXX-XXXX-XXXXXXXXXXXX AccessDeniedException; proxy: null)

Solución: en la integración del catálogo del trabajo de AWS Glue, AWS Glue siempre intenta comprobar si la base de datos predeterminada existe o no utiliza AWS Glue `GetDatabase API`. Cuando no se concede el permiso `DESCRIBE` de Lake Formation o se concede el permiso `GetDatabase IAM`, el trabajo falla al verificar la existencia de la base de datos predeterminada.

Para resolverlo:

1. Agregue el permiso `DESCRIBE` en Lake Formation para la base de datos predeterminada.
2. Configure el rol de IAM asociado al trabajo de AWS Glue como creador de bases de datos en Lake Formation. Esto creará automáticamente una base de datos predeterminada y otorgará los permisos de Lake Formation necesarios para el rol.
3. Deshabilite la opción `--enable-data-catalog`. (Se muestra como Usar Data Catalog como metaalmacén de Hive en AWS Glue Studio).

   Si no necesita la integración Data Catalog de Spark SQL en el trabajo, puede desactivarlo.

Mensaje de error: el mapa de teclas proporcionado no es adecuado para determinados marcos de datos

Condición de error: el mapa de teclas dado no es adecuado para determinados marcos de datos.

Solución: está utilizando el tipo de regla y las claves de unión están duplicadas. DataSetMatch Las claves de unión deben ser únicas y no deben ser NULAS. En los casos en los que no pueda tener claves de combinación que sean únicas, considere la posibilidad de utilizar otros tipos de reglas, como AggregateMatchlas que coincidan en los datos resumidos.

Excepción en la clase de usuario: java.lang. RuntimeException : No se pudieron recuperar los datos. Compruebe los inicios de sesión CloudWatch para obtener más detalles

Condición de error: excepción en la clase de usuario: java.lang. RuntimeException : No se pudieron recuperar los datos. Comprueba los inicios de sesión CloudWatch para obtener más información.

Solución: Esto ocurre cuando crea reglas de DQ en una tabla basada en Amazon S3 que se compara con Amazon RDS o. Amazon Redshift En estos casos, AWS Glue no puede cargar la conexión. En su lugar, intente configurar la regla DQ en el conjunto de datos Amazon Redshift o en Amazon RDS. Este es un error conocido.

ERROR DE INICIO: error al descargar desde S3 para bucket

Condición de error: ERROR DE INICIO: error durante la descarga desde S3 para bucket: `aws-glue-ml-data-quality-assets-us-east-1, key: jars/aws-glue-ml-data-quality-etl.jar.Access Denied (Service: Amazon S3; Status Code: 403; Please refer logs for details)` .

Solución: Los permisos del rol transferidos a AWS Glue Data Quality deben permitir la lectura desde la ubicación anterior de Amazon S3. Esta política de IAM debe estar asociada con la función:

```
{
  "Sid": "allowS3",
  "Effect": "Allow",
  "Action": "s3:GetObject",
  "Resource": "arn:aws:s3:::aws-glue-ml-data-quality-assets-<region>/*"
}
```

Consulte la autorización de calidad de los datos para obtener información detallada sobre los permisos. Estas bibliotecas son necesarias para evaluar la calidad de los datos de sus conjuntos de datos.

InvalidInputException (estado: 400): DataQuality las reglas no se pueden analizar

Condición de error: InvalidInputException (estado: 400): DataQuality las reglas no se pueden analizar.

Solución: hay muchas posibilidades de que se produzca este error. Una posibilidad es que las reglas contengan comillas simples. Revise que sean comillas dobles. Por ejemplo:

```
Rules = [
ColumnValues "tipo_vinculo" in ["CODO", "DOCO", "COCO", "DODO"] AND "categoria" = 'ES'
    AND "cod_bandera" = 'CEP'
```

Cambie esto a:

```
Rules = [
(ColumnValues "tipovinculo" in [ "CODO", "DOCO", "COCO", "DODO"]) AND (ColumnValues
 "categoria" = "ES")
    AND (ColumnValues "codbandera" = "CEP")
]
```

Error: EventBridge no activa los trabajos de calidad de datos de Glue según la programación que configuré

Condición de error: EventBridge no activa trabajos de AWS Glue Data Quality según la programación que configuré.

Solución: es posible que el rol que activa el trabajo no tenga los permisos necesarios. Asegúrese de que el rol que usa para activar los trabajos tenga los permisos necesarios mencionados en la configuración de IAM necesaria para programar las ejecuciones de evaluación.

Errores de CustomSQL

Condición de error: `The output from CustomSQL must contain at least one column that matches the input dataset for AWS Glue Data Quality to provide row level results. The SQL query is a valid query but no columns from the SQL result are present in the Input Dataset. Ensure that matching columns are returned from the SQL.`

Solución: la consulta SQL es válida, pero asegúrese de seleccionar únicamente las columnas de la tabla principal. Se puede producir este error cuando selecciona funciones de agregación como sumar o contar en las columnas de la tabla principal.

Condición de error: `There was a problem when executing your SQL statement: cannot resolve "Col"`.

Solución: esta columna no está presente en la tabla principal.

Condición de error: `The columns that are returned from the SQL statement should only belong to the primary table. "In this case, some columns ( Col ) belong to reference table"`.

Solución: en las consultas SQL, cuando una la tabla principal con otras tablas de referencia, asegúrese de que la instrucción de selección solo contenga los nombres de las columnas de la tabla principal para generar resultados a nivel de fila para la tabla principal.

Reglas dinámicas

Condición de error: `Dynamic rules require job context, and cannot be evaluated in interactive session or data preview.`.

Causa: es posible que este mensaje de error aparezca en los resultados de la vista previa de datos o en otras sesiones interactivas cuando hay reglas de calidad de datos dinámicas en el conjunto de reglas. Las reglas dinámicas hacen referencia a métricas históricas asociadas con un nombre de trabajo y un contexto de evaluación determinados, por lo que no se pueden evaluar en sesiones interactivas.

Solución: Al ejecutar el trabajo de AWS Glue se generarán métricas históricas, a las que se podrá hacer referencia en futuras ejecuciones del mismo trabajo.

Condición de error:

- `[RuleType] rule only supports simple atomic operands in thresholds.`.
- `Function last not yet implemented for [RuleType] rule.`

Resolución: Por lo general, se aceptan reglas dinámicas para todos los tipos de reglas de DQDL en expresiones numéricas (consulte la referencia sobre DQDL). Sin embargo, algunas reglas que generan múltiples métricas aún no son compatibles. ColumnValues ColumnLength

Condición de error: `Binary expression operands must resolve to a single number.`.

Causa: las reglas dinámicas aceptan expresiones binarias, como `RowCount > avg(last(5)) * 0.9`. En este caso, la expresión binaria es `avg(last(5)) * 0.9`. Esta regla es válida ya que

ambos operandos, `avg(last(5))` y `0.9`, se resuelven en un solo número. Un ejemplo incorrecto es `RowCount > last(5) * 0.9`, ya que `last(5)` generará una lista que no se puede comparar de manera significativa con el recuento de filas actual.

Solución: use las funciones de agregación para reducir un operando con valores de lista a un solo número.

Condición de error:

- `Rule threshold results in list, and a single value is expected. Use aggregation functions to produce a single value. Valid example: sum(last(10)), avg(last(10)).`
- `Rule threshold results in empty list, and a single value is expected.`

Causa: es posible usar las reglas dinámicas para comparar alguna característica del conjunto de datos con sus valores históricos. La última función permite recuperar varios valores históricos, siempre que se dé un argumento entero positivo. Por ejemplo, `last(5)` recuperará los cinco últimos valores que se observaron en las ejecuciones de tareas de su regla.

Solución: es necesario utilizar una función de agregación para reducir estos valores a un solo número, para poder realizar una comparación significativa con el valor observado en la ejecución de tareas actual.

Ejemplos válidos:

- `RowCount >= avg(last(5))`
- `RowCount > last(1)`
- `RowCount < last()`

Ejemplo no válido: `RowCount > last(5)`.

Condición de error:

- `Function index used in threshold requires positive integer argument.`
- `Index argument must be an integer. Valid syntax example: RowCount > index(last(10, 2)), which means RowCount must be greater than third most recent execution from last 10 job runs.`

Solución: al crear reglas dinámicas, puede usar la función de agregación `index` para seleccionar un valor histórico de una lista. Por ejemplo, `RowCount > index(last(5)`: 1) comprobará si el recuento de filas observado en el trabajo actual es estrictamente superior al segundo recuento de filas más reciente observado en su trabajo. `index` está indexado a cero.

Condición de error: `IllegalArgumentException: Parsing Error: Rule Type: DetectAnomalies is not valid`.

Resolución: la detección de anomalías solo está disponible en la versión 4.0. de AWS Glue.

Condición de error: `IllegalArgumentException: Parsing Error: Unexpected condition for rule of type ... no viable alternative at input ...`.

Nota: `...` es dinámico. Ejemplo:`IllegalArgumentException: Parsing Error: Unexpected condition for rule of type RowCount with number return type, line 4:19 no viable alternative at input '>last'`.

Resolución: la detección de anomalías solo está disponible en la versión 4.0. de AWS Glue.

Excepción en la clase de usuario: org.apache.spark.sql. AnalysisException: org.apache.hadoop.hive.ql.metadata. HiveException

Condición del error: `Exception in User Class: org.apache.spark.sql.AnalysisException: org.apache.hadoop.hive.ql.metadata.HiveException: Unable to fetch table mailpiece_submitted. StorageDescriptor#InputFormat cannot be null for table: mailpiece_submitted (Service: null; Status Code: 0; Error Code: null; Request ID: null; Proxy: null)`

Causa: está utilizando Apache Iceberg en el catálogo de datos de AWS Glue y el atributo Formato de entrada del catálogo de datos de AWS Glue está vacío.

Solución: este problema se produce cuando se utiliza el tipo de regla CustomSQL en la regla de calidad de datos. Una forma de solucionarlo es utilizar “primario” o agregar el nombre del catálogo `glue_catalog.` a `<database>.<table> in Custom ruletype`.

UNCLASSIFIED_ERROR; IllegalArgumentException: Error de análisis: no se proporcionaron reglas ni analizadores., no hay una alternativa viable en la entrada

Condición del error: `UNCLASSIFIED_ERROR; IllegalArgumentException: Parsing Error: No rules or analyzers provided., no viable alternative at input`

Resolución: el DQDL no se puede analizar. Hay algunos casos en los que esto puede ocurrir. Si utiliza reglas compuestas, asegúrese de que estén entre paréntesis.

```
(RowCount >= avg(last(10)) * 0.6) and (RowCount <= avg(last(10)) * 1.4) instead of
      RowCount >= avg(last(10)) * 0.6 and RowCount <= avg(last(10)) * 1.4
```

Integración de datos de Amazon Q en AWS Glue

La integración de datos de Amazon Q AWS Glue es una nueva capacidad de IA generativa AWS Glue que permite a los ingenieros de datos y a los desarrolladores de ETL crear trabajos de integración de datos utilizando un lenguaje natural. Los ingenieros y desarrolladores pueden pedir a Amazon Q que cree trabajos, solucione problemas y responda a preguntas sobre AWS Glue la integración de datos.

¿Qué es Amazon Q?

> **Note**
>
> Desarrollado por Amazon Bedrock: AWS implementa la detección automática de abusos. Gracias a que la integración de datos de Amazon Q está integrada en Amazon Bedrock, los usuarios pueden aprovechar al máximo los controles implementados en Amazon Bedrock para reforzar la protección, la seguridad y el uso responsable de la inteligencia artificial (IA).

Amazon Q es un asistente conversacional basado en inteligencia artificial (IA) generativa que puede ayudarlo a comprender, crear, ampliar y operar AWS aplicaciones. El modelo en el que se basa Amazon Q se ha ampliado con AWS contenido de alta calidad para ofrecerte respuestas más completas, procesables y referenciadas que te permitan acelerar tu desarrollo. AWS Para obtener más información, consulte ¿Qué es Amazon Q?

¿En qué consiste la integración de datos de Amazon Q en AWS Glue?

La integración de datos de Amazon Q AWS Glue incluye las siguientes capacidades:

- Chat: la integración de datos de Amazon Q AWS Glue puede responder a preguntas en inglés en lenguaje natural sobre AWS Glue los dominios de integración de datos, como los conectores de AWS Glue origen y destino, los trabajos de AWS Glue ETL, el catálogo de datos, los rastreadores y AWS Lake Formation otras funciones, la documentación y las mejores prácticas. La integración de datos de Amazon Q AWS Glue responde a step-by-step instrucciones e incluye referencias a sus fuentes de información.

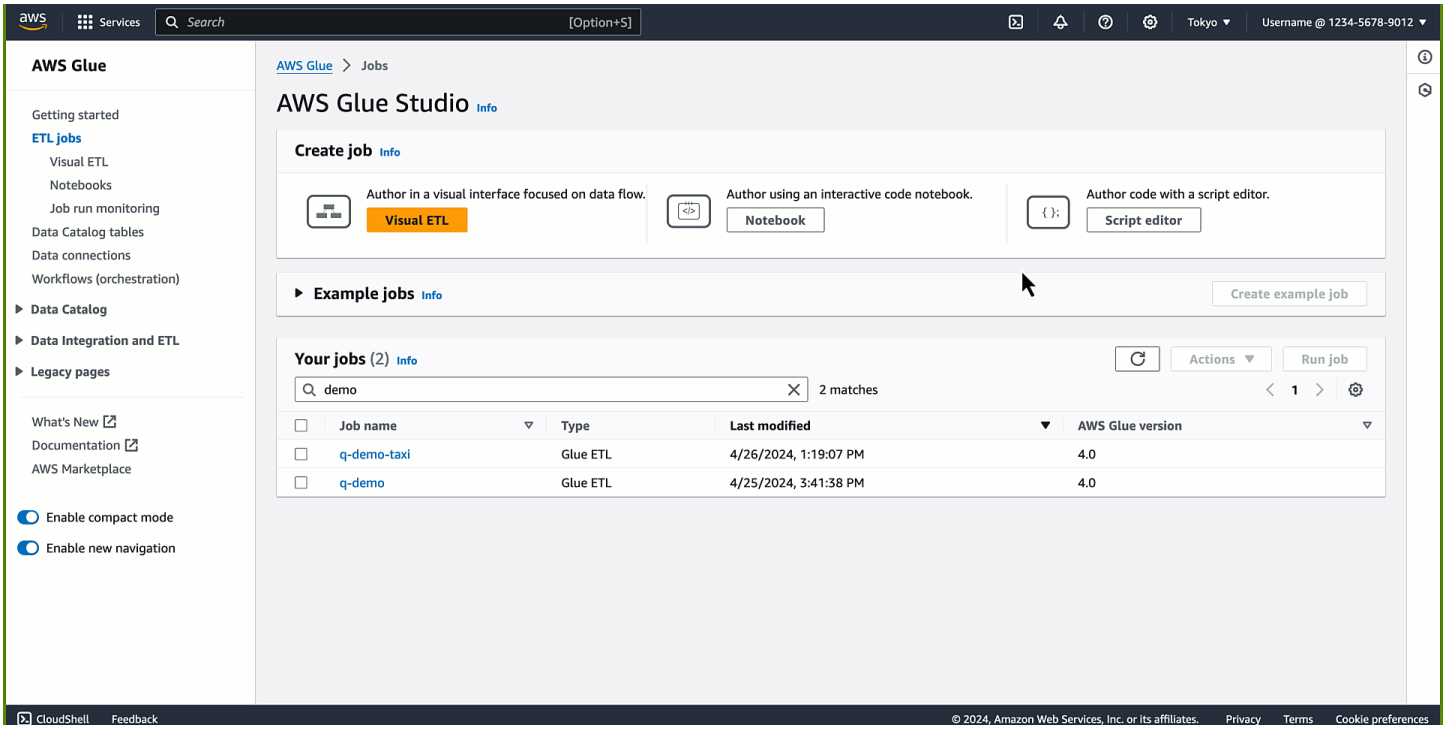
- Generación de código de integración de datos: la integración de datos de Amazon Q AWS Glue puede responder a preguntas sobre los scripts de AWS Glue ETL y generar código nuevo si se responde a una pregunta en inglés en lenguaje natural.
- Solución de problemas: la integración de datos de Amazon Q AWS Glue está diseñada específicamente para ayudarlo a comprender los errores en los AWS Glue trabajos y proporciona step-by-step instrucciones para identificar la causa y resolver sus problemas.

> **Note**
>
> La integración de datos de Amazon Q AWS Glue no utiliza el contexto de la conversación para generar respuestas futuras mientras dure la conversación. Cada conversación con la integración de datos de Amazon Q AWS Glue es independiente de sus conversaciones anteriores o futuras.

¿Está trabajando con la integración de datos de Amazon Q en AWS Glue?

En el panel de Amazon Q, puedes solicitar a Amazon Q que genere el código para un script de AWS Glue ETL o responder a una pregunta sobre AWS Glue las funciones o la solución de un error. La respuesta es un script ETL PySpark con step-by-step instrucciones para personalizar el script, revisarlo y ejecutarlo. Para las preguntas, la respuesta se genera a partir de la base de conocimientos sobre integración de datos, con un resumen y una URL fuente como referencia.

Por ejemplo, puedes pedirle a Amazon Q que diga «Por favor, proporciona un script de Glue que lea de Snowflake, cambie el nombre de los campos y escriba en Redshift» y, en respuesta, la integración de datos de Amazon Q AWS Glue devolverá un script de AWS Glue trabajo que puede realizar la acción solicitada. Puede revisar el código generado para asegurarse de que cumple la intención solicitada. Si está satisfecho, puede implementarlo como un AWS Glue trabajo en producción. Puede solucionar los problemas de los trabajos pidiendo a la integración que explique los errores y fallos y que proponga soluciones. Amazon Q puede responder a las preguntas sobre las prácticas recomendadas de integración de datos AWS Glue o a las prácticas recomendadas para la integración de datos.



Las siguientes son preguntas de ejemplo que demuestran cómo la integración de datos de Amazon Q AWS Glue puede ayudarlo a desarrollar AWS Glue:

AWS Glue Generación de código ETL:

- Escriba un AWS Glue script que lea JSON de S3, transforme los campos mediante la asignación de aplicaciones y escriba en Amazon Redshift
- ¿Cómo escribo un AWS Glue script para leer desde DynamoDB, aplicar DropNullFields la transformación y escribir en S3 como Parquet?
- Deme un AWS Glue script que lea desde MySQL, elimine algunos campos según mi lógica empresarial y escriba en Snowflake
- Escriba un AWS Glue trabajo para leerlo desde DynamoDB y escriba en S3 como JSON
- Ayúdeme a desarrollar un AWS Glue script para el catálogo de AWS Glue datos para S3
- Escribe un AWS Glue trabajo para leer JSON desde S3, elimina los nulos y escribe en Redshift

AWS Glue explicaciones de las funciones:

- ¿Cómo utilizo la calidad AWS Glue de los datos?
- ¿Cómo usar los marcadores de AWS Glue trabajo?
- ¿Cómo activo el ajuste de escala AWS Glue automático?

- ¿Cuál es la diferencia entre los marcos AWS Glue dinámicos y los marcos de datos de Spark?
- ¿Cuáles son los diferentes tipos de conexiones compatibles AWS Glue?

AWS Glue solución de problemas:

- ¿Cómo solucionar los errores de falta de memoria (OOM) en AWS Glue los trabajos?
- ¿Cuáles son algunos de los mensajes de error que puede ver al configurar la calidad de AWS Glue los datos y cómo puede solucionarlos?
- ¿Cómo soluciono un AWS Glue trabajo con el error Acceso denegado a Amazon S3?
- ¿Cómo soluciono los problemas relacionados con la reorganización de datos en los AWS Glue trabajos?

Prácticas recomendadas para interactuar con la integración de datos de Amazon Q

Las siguientes son las prácticas recomendadas para interactuar con la integración de datos de Amazon Q:

- Cuando interactúe con la integración de datos de Amazon Q, formule preguntas específicas, repita cuando tenga solicitudes complejas y compruebe la precisión de las respuestas.
- Cuando proporcione instrucciones de integración de datos en lenguaje natural, sea lo más específico posible para ayudar al asistente a entender exactamente lo que necesita. En lugar de preguntar «extraiga datos de S3», proporcione más detalles, como «escriba un AWS Glue script que extraiga archivos JSON de S3».
- Revise el script generado antes de ejecutarlo para garantizar su precisión. Si el script generado contiene errores o no coincide con su intención, dé instrucciones al asistente sobre cómo corregirlo.
- La tecnología de IA generativa es nueva y puede haber errores (a veces denominados alucinaciones) en las respuestas. Pruebe y revise todo el código para detectar errores y vulnerabilidades antes de usarlo en su entorno o carga de trabajo.

Integración de datos de Amazon Q en la mejora AWS Glue del servicio

Para ayudar a la integración de datos de Amazon Q a AWS Glue proporcionar la información más relevante sobre AWS los servicios, podemos utilizar cierto contenido de Amazon Q, como las preguntas que le haces a Amazon Q y sus respuestas, para mejorar el servicio.

Para obtener información sobre el contenido que utilizamos y cómo excluirlo, consulta la mejora del servicio Amazon Q Developer en la Guía del usuario para desarrolladores de Amazon Q.

Consideraciones

Tenga en cuenta los siguientes aspectos antes de utilizar la integración de datos de Amazon Q en AWS Glue:

- Actualmente, la generación de código solo funciona con PySpark el kernel. El código generado es para AWS Glue trabajos basados en Python Spark.
- Para obtener información sobre las combinaciones compatibles de capacidades de generación de código de la integración de datos de Amazon Q en AWS Glue, consulteCapacidades de generación de código compatibles.

Configuración de la integración de datos de Amazon Q en AWS Glue

En las siguientes secciones se ofrece información sobre la configuración de la integración de datos de Amazon Q en AWS Glue.

Temas

- Configuración de permisos de IAM

Configuración de permisos de IAM

En este tema se describen los permisos de IAM que se configuran para la experiencia de chat de Amazon Q y la experiencia de AWS Glue Studio Notebook.

Temas

- Configuración de los permisos de IAM para el chat de Amazon Q
- Configuración de los permisos de IAM para las libretas Studio AWS Glue

Configuración de los permisos de IAM para el chat de Amazon Q

La concesión de permisos a las API utilizadas por la integración de datos de Amazon Q AWS Glue requiere los permisos de AWS Identity and Access Management (IAM) adecuados. Puede obtener permisos adjuntando la siguiente AWS política personalizada a su identidad de IAM (por ejemplo, un usuario, un rol o un grupo):

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:StartCompletion",
                "glue:GetCompletion"
            ],
            "Resource": [
                "arn:aws:glue:*:*:completion/*"
            ]
        }
    ]
}
```

Configuración de los permisos de IAM para las libretas Studio AWS Glue

Para habilitar la integración de datos de Amazon Q en las libretas de AWS Glue Studio, asegúrate de que el siguiente permiso esté asociado a la función de IAM de libretas:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:StartCompletion",
                "glue:GetCompletion"
            ],
```

```
            "Resource": [
                "arn:aws:glue:*:*:completion/*"
            ]
        },
        {
            "Sid": "CodeWhispererPermissions",
            "Effect": "Allow",
            "Action": [
                "codewhisperer:GenerateRecommendations"
            ],
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> La integración de datos de Amazon Q AWS Glue no tiene API disponibles a través del AWS SDK que pueda usar mediante programación. Las dos API siguientes se utilizan en la política de IAM para permitir esta experiencia a través del panel de chat de Amazon Q o los cuadernos de AWS Glue Studio: `StartCompletion` y. `GetCompletion`

Asignación de permisos

Para dar acceso, agregue permisos a los usuarios, grupos o roles:

- Usuarios y grupos en el Centro de Identidad de AWS IAM: cree un conjunto de permisos. Siga las instrucciones que se detallan en Creación de un conjunto de permisos en la Guía del usuario de AWS IAM Identity Center.
- Usuarios administrados en IAM a través de un proveedor de identidades: Creación de un rol para la federación de identidades. Siga las instrucciones descritas en Creación de un rol para un proveedor de identidad de terceros (federación) en la Guía del usuario de IAM.
- Usuarios de IAM:
  - Cree un rol que el usuario pueda aceptar. Siga las instrucciones descritas en Creación de un rol para un usuario de IAM en la Guía del usuario de IAM.
  - (No recomendado) Adjunte una política directamente a un usuario o añada un usuario a un grupo de usuarios. Siga las instrucciones descritas en Adición de permisos a un usuario (consola) de la Guía del usuario de IAM.

Capacidades de generación de código compatibles

A continuación se muestran las combinaciones de las capacidades de generación de código de la integración de datos de Amazon Q en AWS Glue.

Origen	Transformación	Destino
S3 con los siguientes tipos de formato: json, csv, parquet, hudi, delta	ApplyMapping	S3 con los siguientes tipos de formato: json, csv, avro, orc, parquet, hudi, delta
Glue Data Catalog	RenameField	Catálogo de datos de Glue
Amazon Redshift	DropFields	Amazon Redshift
MySQL	SelectFields	MySQL
Postgres	DropNullFields	Postgres
Oracle	Filtro	Oracle
SQL Server	Spigot	SQL Server
DynamoDB	Código SQL personalizado	DynamoDB
Snowflake	Agregado	Snowflake
MongoDB	DropDuplicates	MongoDB
Conector JDBC personalizado	Join	Conector JDBC personalizado
Conector Spark personalizado	Unión	Conector Spark personalizado
Google BigQuery		Google BigQuery
Teradata		Teradata
OpenSearch Servicio Amazon		OpenSearch Servicio Amazon
Vertica		Vertica

Origen	Transformación	Destino
Azure SQL		Azure DQL
SAP HANA		SAP HANA
Azure Cosmos		Cosmos azul

Interacciones de muestra

La integración de datos de Amazon Q te AWS Glue permite introducir tu pregunta en el panel de Amazon Q. Puede ingresar una pregunta sobre la funcionalidad de la integración de datos de AWS Glue. Obtendrá una respuesta detallada, junto con los documentos de referencia.

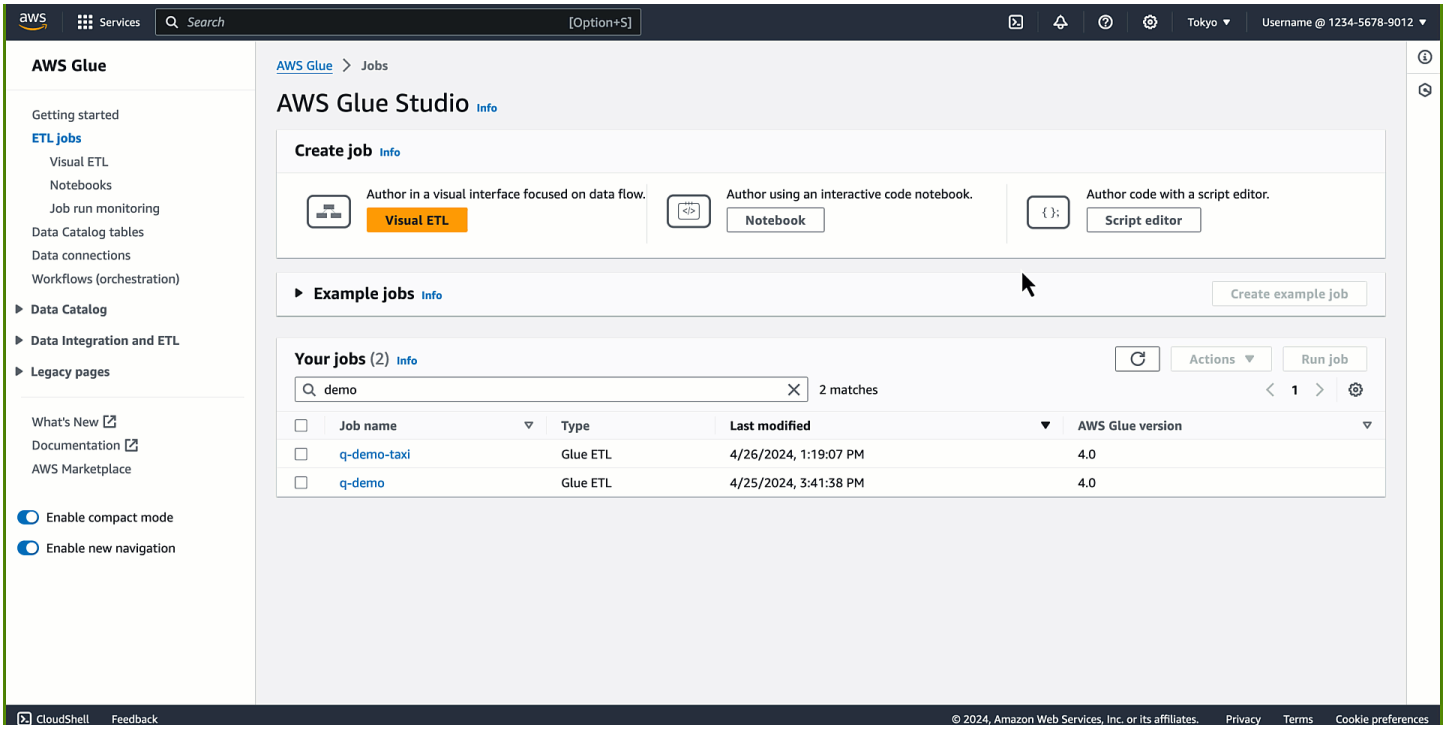
Otro caso de uso es la generación de scripts de trabajo de AWS Glue ETL. Puede preguntar cómo ejecutar un trabajo de extracción, transformación y carga de datos. Se devolverá un PySpark script generado.

Temas

- Interacciones de chat de Amazon Q
- AWS Glue Interacciones de Studio Notebook

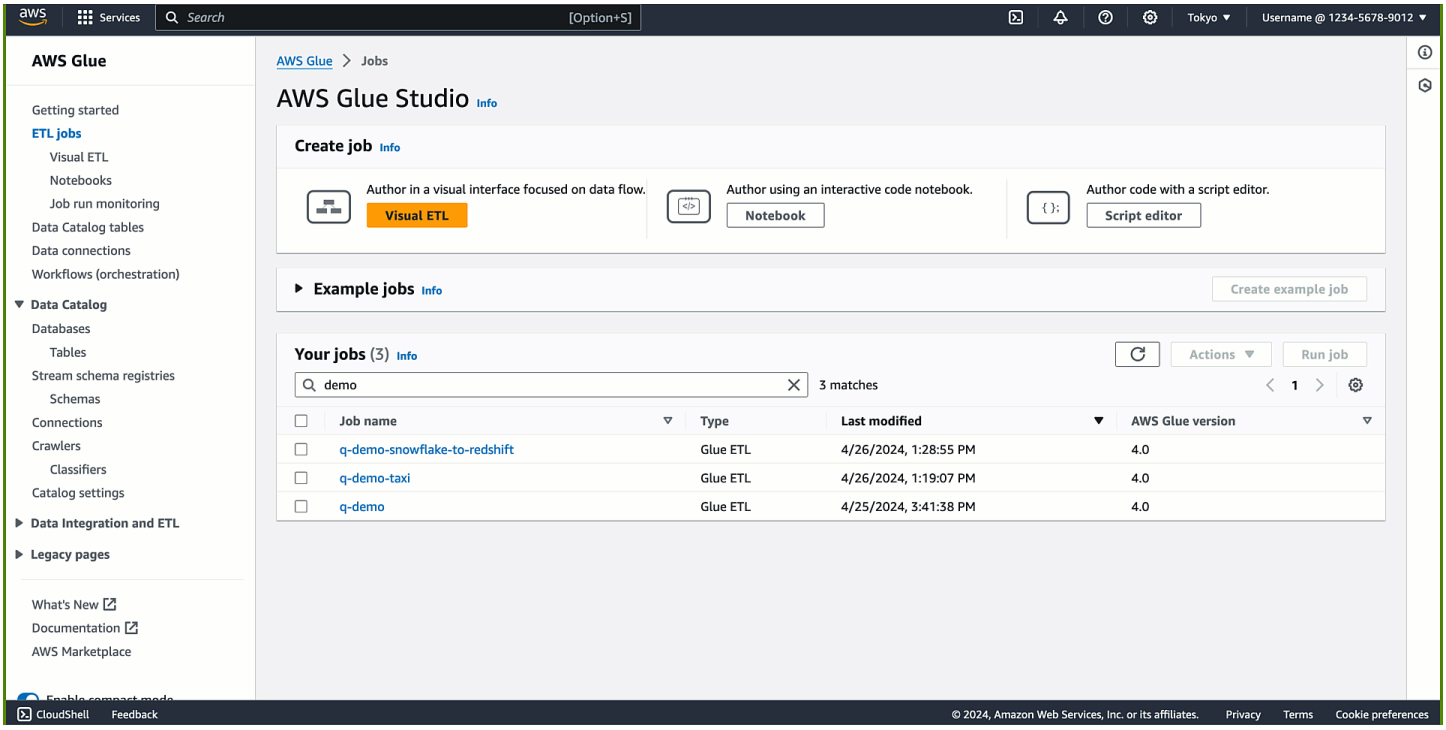
Interacciones de chat de Amazon Q

En la AWS Glue consola, comience a crear un nuevo trabajo y pregunte a Amazon Q: «Proporcione un script de Glue que lea Snowflake, cambie el nombre de los campos y escriba en Redshift».



Observará que el código está generado. Con esta respuesta, puede aprender y comprender cómo crear AWS Glue código para su propósito. Puede copiar y pegar el código generado en el editor de scripts y configurar los marcadores de posición. Después de configurar un rol de AWS Identity and Access Management (IAM) y AWS Glue las conexiones del trabajo, guárdelo y ejecútelo. Cuando finalice el trabajo, puede empezar a consultar la tabla exportada desde Snowflake en Amazon Redshift.

El siguiente mensaje lee los datos de dos fuentes diferentes, los filtra y proyecta de forma individual, los une en una clave común y escribe el resultado en un tercer destino. Pregúntele a Amazon Q: «Quiero leer datos de S3 en formato Parquet y seleccionar algunos campos. También quiero leer datos de DynamoDB, seleccionar algunos campos y filtrar algunas filas. Quiero unir estos dos conjuntos de datos y escribir los resultados en ellos. OpenSearch

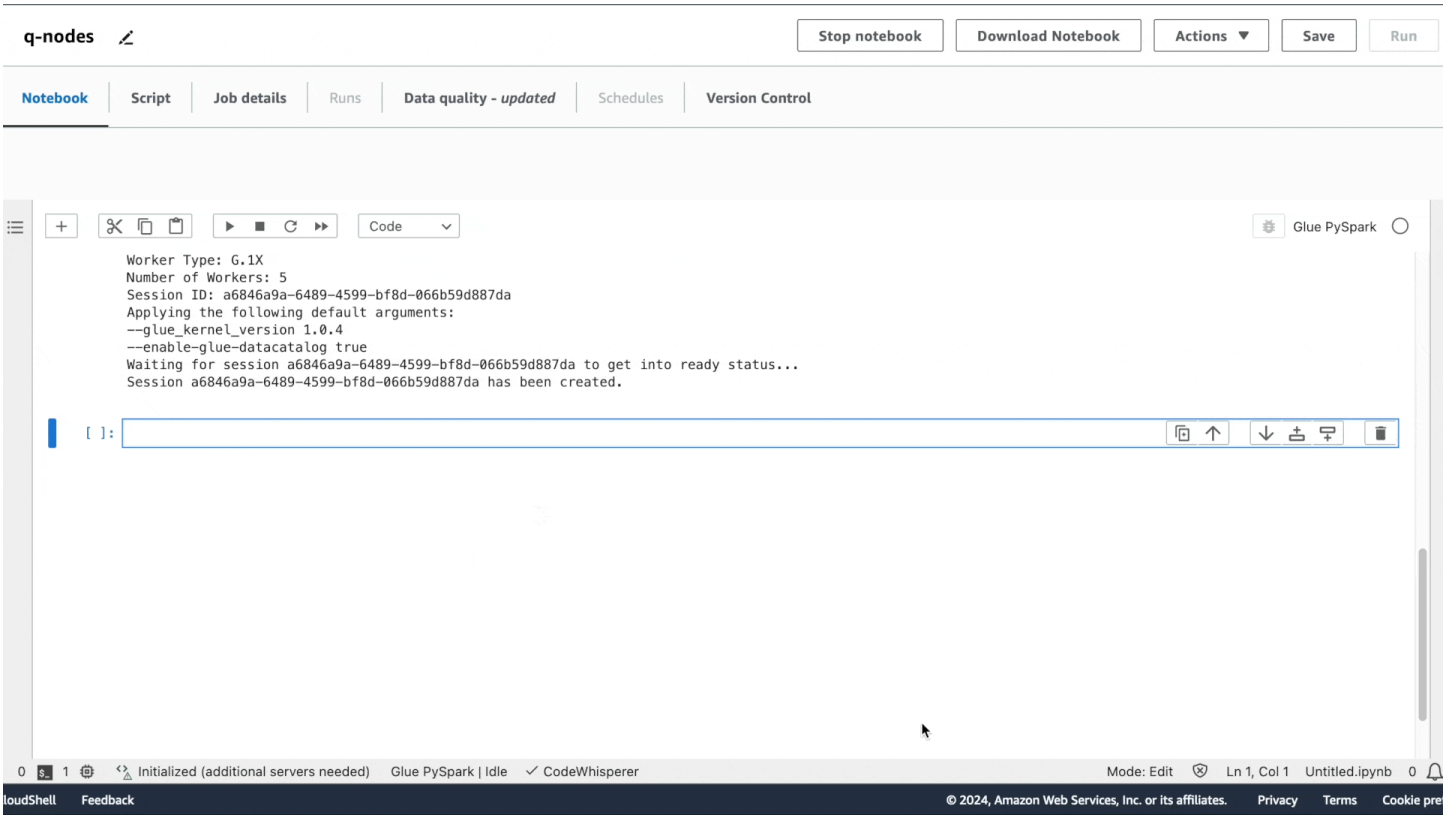


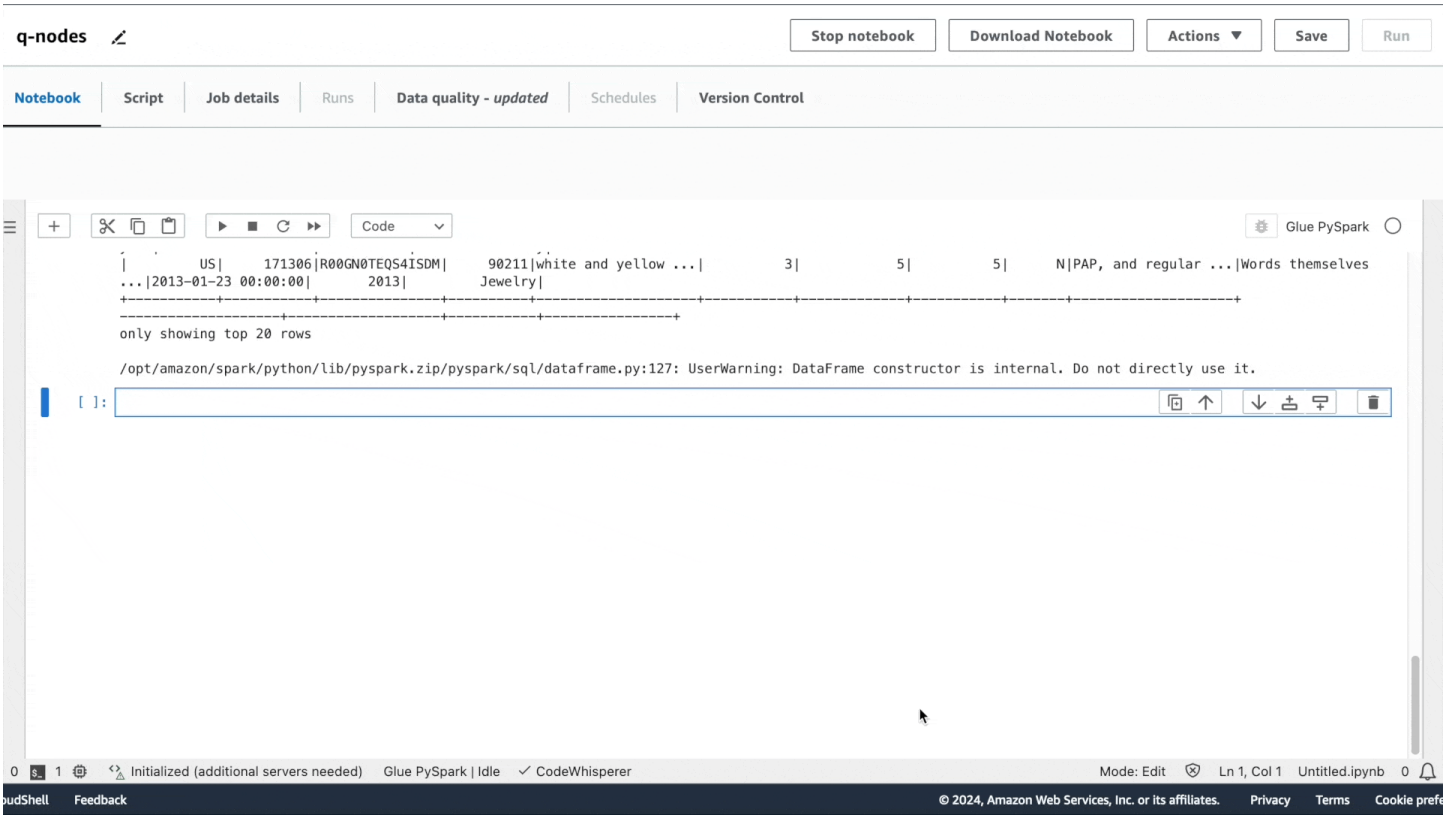
Se genera el código. Cuando se complete el trabajo, su índice estará disponible en sus cargas de trabajo posteriores OpenSearch y lo podrán utilizar.

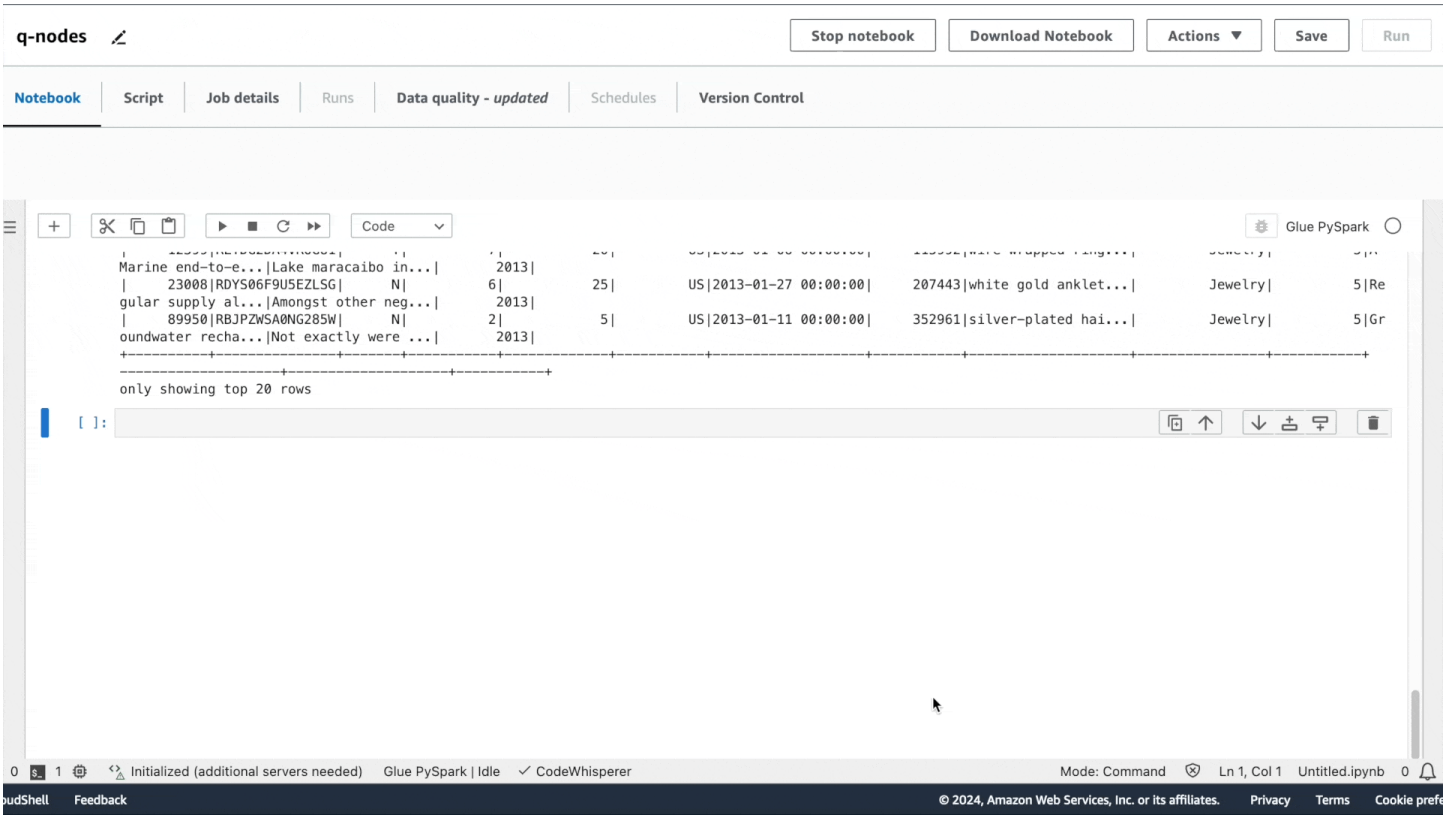
AWS Glue Interacciones de Studio Notebook

Añada una nueva celda e introduzca su comentario para describir lo que quiere conseguir. Después de pulsar la tecla Tab y Entrar, se muestra el código recomendado.

La primera intención es extraer los datos: «Dame un código que lea una tabla del catálogo de datos de Glue», seguido de «Dame un código para aplicar una transformación de filtro con star_rating>3" y «Dame un código que escriba el marco en S3 como Parquet».


q-nodes
Stop notebook
Download Notebook
Actions
Save
Run
Notebook
Script
Job details
Runs
Data quality - updated
Schedules
Version Control
Code
Glue PySpark
Worker Type: G.1X
Number of Workers: 5
Session ID: a6846a9a-6489-4599-bf8d-066b59d887da
Applying the following default arguments:
--glue_kernel_version 1.0.4
--enable-glue-datacatalog true
Waiting for session a6846a9a-6489-4599-bf8d-066b59d887da to get into ready status...
Session a6846a9a-6489-4599-bf8d-066b59d887da has been created.
[ ]:
Initialized (additional servers needed)
Glue PySpark | Idle
CodeWhisperer
Mode: Edit
Ln 1, Col 1
Untitled.ipynb
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms


q-nodes
Stop notebook
Download Notebook
Actions
Save
Run
Notebook
Script
Job details
Runs
Data quality - updated
Schedules
Version Control
Code
Glue PySpark
| US| 171306|R00GN0TEQS4ISDM| 90211|white and yellow ...| 3| 5| 5| N|PAP, and regular ...|Words themselves
...|2013-01-23 00:00:00| 2013| Jewelry|
only showing top 20 rows
/opt/amazon/spark/python/lib/pyspark.zip/pyspark/sql/dataframe.py:127: UserWarning: DataFrame constructor is internal. Do not directly use it.
[ ]:
Initialized (additional servers needed)
Glue PySpark | Idle
CodeWhisperer
Mode: Edit
Ln 1, Col 1
Untitled.ipynb
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms

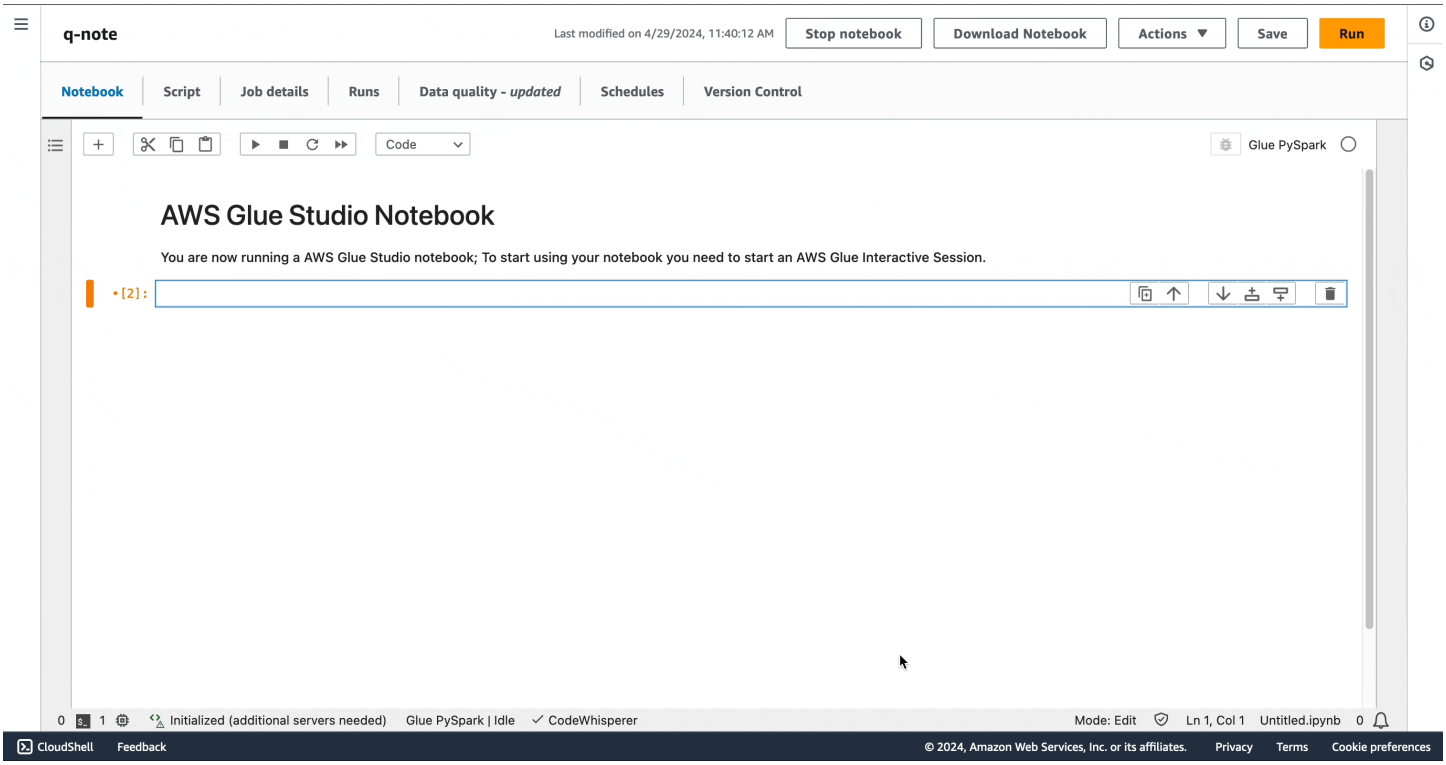


Al igual que en la experiencia de chat de Amazon Q, se recomienda usar el código. Si presionas la tecla Tab, se elige el código recomendado.

Puede ejecutar cada celda rellenando las opciones apropiadas para sus fuentes en el código generado. En cualquier momento de las ejecuciones, también puedes obtener una vista previa de una muestra de tu conjunto de datos mediante el `show( )` método.

Indicaciones complejas

Puede generar un guion completo con un único mensaje complejo. «Tengo datos de JSON en S3 y datos de Oracle que es necesario combinar. Proporcione un script de Glue que lea ambas fuentes, realice una unión y, a continuación, escriba los resultados en Redshift».



Puede observar que, en la libreta, la integración de datos de Amazon Q AWS Glue generó el mismo fragmento de código que se generó en el chat de Amazon Q.

Puede ejecutar el bloc de notas como un trabajo, ya sea seleccionando Ejecutar o mediante programación.

Orquestación en AWS Glue

En las siguientes secciones, se ofrece información acerca de la orquestación de los trabajos en AWS Glue.

Temas

- Inicio de trabajos y rastreadores mediante desencadenadores
- Realización de actividades de ETL complejas mediante esquemas y flujos de trabajo en AWS Glue
- Desarrollo de esquemas en AWS Glue

Inicio de trabajos y rastreadores mediante desencadenadores

En AWS Glue, puede crear objetos del Data Catalog denominados desencadenadores, que puede utilizar para iniciar en forma manual o automática uno o más rastreadores o trabajos de extracción, transformación y carga (ETL). Mediante los desencadenadores, puede diseñar una cadena de trabajos y rastreadores dependientes.

> **Note**
>
> Esto mismo se puede conseguir definiendo flujos de trabajo. Los flujos de trabajo son el método preferido para crear operaciones ETL multitarea complejas. Para obtener más información, consulte the section called “Realización de actividades de ETL complejas mediante esquemas y flujos de trabajo”.

Temas

- Disparadores de AWS Glue
- Agregado de desencadenadores
- Activación y desactivación de desencadenadores

Disparadores de AWS Glue

Cuando está activo, un desencadenador puede iniciar trabajos y rastreadores especificados. Un desencadenador se activa bajo demanda, en función de un calendario o en función de una combinación de eventos.

> **Note**
>
> Solo se pueden habilitar dos rastreadores con un solo desencadenador. Si desea rastrear varios almacenes de datos, utilice múltiples orígenes para cada rastreador en lugar de ejecutar varios rastreadores en forma simultánea.

Un desencadenador puede tener uno de varios estados. Los estados de un desencadenador son `CREATED`, `ACTIVATED` o `DEACTIVATED`. También hay estados de transición, como `ACTIVATING`. Para detener temporalmente la activación de un desencadenador, puede desactivarlo. Podrá reactivarlo más adelante.

Existen tres tipos de desencadenadores:

Programados

Un desencadenador con límite de tiempo basado en `cron`.

Puede crear un desencadenador para un conjunto de trabajos o rastreadores en función de un calendario. Puede especificar restricciones, como la frecuencia de ejecución de los trabajos o rastreadores, qué días de la semana se ejecutan y a qué hora. Estas restricciones se basan en `cron`. Al configurar un calendario para un desencadenador, debe tener en cuenta las características y limitaciones de cron. Por ejemplo, si decide ejecutar su rastreador el día 31 de cada mes, tenga en cuenta que algunos meses no tienen 31 días. Para obtener más información acerca de Cron, consulte Programaciones basadas en tiempo para trabajos y rastreadores.

Condicional

Desencadenador que se activa cuando un trabajo o rastreador anterior o varios trabajos o rastreadores satisfacen una lista de condiciones.

Cuando se crea un desencadenador condicional, se especifica una lista de trabajos y una lista de rastreadores que se deben vigilar. Para cada trabajo o rastreador vigilado, se especifica un estado de vigilancia, como éxito, error, tiempo de espera agotado, etc. El desencadenador se activa si los trabajos o los rastreadores observados terminan con los estados especificados. Puede configurar el desencadenador para que se active cuando se produzca alguno o todos los eventos observados.

Por ejemplo, podría configurar un desencadenador D1 para iniciar el trabajo T3 cuando tanto el trabajo T1 como el trabajo T2 se completen correctamente, y otro desencadenador D2 para iniciar el trabajo T4 si se produce un error en el trabajo T1 o en el trabajo T2.

En la tabla siguiente se enumeran los estados de finalización del trabajo y del rastreador (eventos) que los desencadenadores vigilan.

Estados de finalización del trabajo	Estados de finalización del rastreador
• `SUCCEEDED` • `STOPPED` • `FAILED` • `TIMEOUT`	• `SUCCEEDED` • `FAILED` • `CANCELLED`

Bajo demanda

Un desencadenador que se pone en marcha cuando se activa. Los desencadenadores bajo demanda nunca alcanzan el estado `DEACTIVATED` o `ACTIVATED`. Siempre permanecen en el estado `CREATED`.

Para que estén listos para activarse en cuanto se creen, puede configurar un indicador para activar los desencadenadores programados y condicionales en el momento de crearlos.

> **Important**
>
> Los trabajos o rastreadores que se ejecutan como resultado de la finalización de otros trabajos o rastreadores se denominan dependientes. Los trabajos o los rastreadores dependientes solo se inician si el trabajo o el rastreador que se completa se inició con un desencadenador. Todos los trabajos o rastreadores de una cadena de dependencia deben ser descendientes de un único desencadenador programado o bajo demanda.

Pasar parámetros de trabajo con desencadenadores

Un desencadenador puede pasar parámetros a los trabajos que inicia. Los parámetros incluyen argumentos de trabajo, valor de tiempo de espera y configuración de seguridad, entre otros. Si el desencadenador inicia varios trabajos, los parámetros se pasan a cada trabajo.

Estas son las reglas para los argumentos de trabajo pasados por un desencadenador:

- Si la clave del par de clave-valor coincide con un argumento de trabajo predeterminado, el argumento pasado invalida el argumento predeterminado. Si la clave no coincide con un argumento predeterminado, el argumento se pasa como un argumento adicional al trabajo.
- Si la clave del par de clave-valor coincide con un argumento no invalidable, se omite el argumento pasado.

Para obtener más información, consulte the section called “Desencadenadores” en la API de AWS Glue.

Agregado de desencadenadores

Puede agregar un desencadenador mediante la consola de AWS Glue, la AWS Command Line Interface (AWS CLI) o la API de AWS Glue.

> **Note**
>
> Actualmente, la consola de AWS Glue solo admite trabajos, no rastreadores, cuando se trabaja con desencadenadores. Puede utilizar la AWS CLI o la API de AWS Glue para configurar desencadenadores tanto con trabajos como con rastreadores.

Para agregar un desencadenador (consola)

1. Inicie sesión en la AWS Management Console y abra la consola de AWS Glue en https://console.aws.amazon.com/glue/.
2. En el panel de navegación, en ETL, elija Triggers (Desencadenadores). A continuación, elija Add trigger (Añadir desencadenador).
3. Proporcione las siguientes propiedades:

   Nombre

   Asigne un nombre único al disparador.

   Trigger type (Tipo de disparador)

   Especifique uno de los siguientes valores:

   - Schedule (Calendario): el desencadenador se activa a una frecuencia y hora específicas.

- Job events (Eventos de trabajo): un desencadenador condicional. El desencadenador se activa cuando alguno o todos los trabajos de la lista coinciden con sus estados designados. Para que el desencadenador se active, un desencadenador debe haber iniciado los trabajos vigilados. Para cualquier trabajo que elija, solo puede vigilar un evento de trabajo (estado de finalización).
- On-demand (Bajo demanda): el desencadenador se pone en marcha cuando se activa.

4. Complete el asistente para desencadenadores. En la página Review (Revisar), puede activar inmediatamente los desencadenadores de tipo Schedule (Calendario) y Job events (Eventos de trabajo) (condicional) seleccionando Enable trigger on creation (Habilitar desencadenador al crearlo).

Para añadir un desencadenador (AWS CLI)

- Introduzca un comando similar al siguiente.

```
aws glue create-trigger --name MyTrigger --type SCHEDULED --schedule  "cron(0 12 *
 * ? *)" --actions CrawlerName=MyCrawler --start-on-creation
```

Este comando crea un desencadenador de programación llamado `MyTrigger`, que se ejecuta todos los días a las 12:00 h UTC e inicia un rastreador llamado `MyCrawler`. El desencadenador se crea en el estado activado.

Para obtener más información, consulte the section called “Disparadores de AWS Glue”.

Programaciones basadas en tiempo para trabajos y rastreadores

Puede definir programaciones basadas en tiempo para los rastreadores y los trabajos en AWS Glue. La definición de estas programaciones utiliza sintaxis cron del tipo Unix. Debe utilizar el formato de hora universal coordinada (UTC) y la precisión mínima para una programación es 5 minutos.

Para obtener más información sobre cómo configurar trabajos y rastreadores para que se ejecuten en forma programada, consulte Inicio de trabajos y rastreadores mediante desencadenadores.

Expresiones cron

Las expresiones Cron tienen seis campos obligatorios, que están separados por un espacio en blanco.

Sintaxis

```
cron(Minutes Hours Day-of-month Month Day-of-week Year)
```

Campos	Valores	Caracteres comodín
Minutos	0–59	, - * /
Horas	0–23	, - * /
Día del mes	1–31	, - * ? / L W
Mes	1–12 o ENE-DIC	, - * /
Día de la semana	1–7 o DOM-SÁB	, - * ? / L
Año	1970-2199	, - * /

Caracteres comodín

- El carácter comodín , (coma) incluye valores adicionales. En el campo `Month`, `JAN,FEB,MAR` incluiría enero, febrero y marzo.
- El carácter comodín - (guion) especifica los intervalos. En el campo `Day`, 1-15 incluiría los días del 1 al 15 del mes especificado.
- El * (asterisco) incluye todos los valores del campo. En el campo `Hours`, * incluiría cada hora.
- El comodín / (barra inclinada) especifica incrementos. En el campo `Minutes`, puede escribir **`1/10`** para especificar cada décimo minuto, empezando desde el primer minuto de la hora (por ejemplo, los minutos 11, 21 y 31, etc.).
- El comodín ? (signo de interrogación) especifica uno u otro. En el campo `Day-of-month` puede escribir 7 y si no le importa qué día de la semana era el séptimo, podría escribir ? en el campo Day-of-week.
- El comodín L en los campos `Day-of-month` o `Day-of-week` especifica el último día del mes o de la semana.
- El comodín W en el campo `Day-of-month` especifica un día de la semana. En el campo `Day-of-month`, `3W` especifica el día más cercano al tercer día de semana del mes.

Límites

- No se pueden especificar los campos `Day-of-month` y `Day-of-week` en la misma expresión Cron. Si especifica un valor en uno de los campos, debe utilizar un ? (signo de interrogación) en el otro.
- No se admiten las expresiones Cron que producen frecuencias superiores a 5 minutos.

Ejemplos

Cuando cree una programación, puede utilizar las siguientes cadenas Cron de ejemplo.

Minutos	Horas	Día del mes	Mes	Día de la semana	Año	Significado
0	10	*	*	?	*	Ejecutar a las 10:00 h (UTC) todos los días
15	12	*	*	?	*	Ejecutar a las 12:15 h (UTC) todos los días
0	18	?	*	MON-FRI	*	Ejecutar a las 18:00 h (UTC) de lunes a viernes
0	8	1	*	?	*	Ejecutar a las 08:00 horas (UTC) todos los

Minutos	Horas	Día del mes	Mes	Día de la semana	Año	Significado
						primeros de mes
0/15	*	*	*	?	*	Ejecutar cada 15 minutos
0/10	*	?	*	MON-FRI	*	Ejecutar cada 10 minutos de lunes a viernes
0/5	8-17	?	*	MON-FRI	*	Ejecutar cada 5 minutos de lunes a viernes entre las 8.00 y las 17.55 h (UTC)

Por ejemplo, para ejecutarse en un programa diario a las 12:15 UTC, especifique:

```
cron(15 12 * * ? *)
```

Activación y desactivación de desencadenadores

Puede activar o desactivar un disparador mediante la AWS Glue consola, el AWS Command Line Interface (AWS CLI) o la AWS Glue API.

Para activar o desactivar un desencadenador (consola)

1. Inicia sesión en la AWS Glue consola AWS Management Console y ábrela en https://console.aws.amazon.com/glue/.
2. En el panel de navegación, en ETL, elija Triggers (Desencadenadores).
3. Active la casilla de verificación situada junto al desencadenador que desee y, en el menú Action (Acción) elija Enable trigger (Activar desencadenador) para activar el desencadenador o Disable trigger (Desactivar desencadenador) para desactivarlo.

Para activar o desactivar un desencadenador (AWS CLI)

- Especifique uno de los siguientes comandos.

```
aws glue start-trigger --name MyTrigger

aws glue stop-trigger --name MyTrigger
```

  Al iniciar un desencadenador se activa y al detenerlo se desactiva. Cuando activa un desencadenador bajo demanda, este se activa inmediatamente.

Para obtener más información, consulte the section called “Disparadores de AWS Glue”.

Realización de actividades de ETL complejas mediante esquemas y flujos de trabajo en AWS Glue

Es posible que algunos de los procesos complejos del servicio ETL (extracción, transformación y carga) de su organización se implementen de mejor manera con múltiples trabajos y rastreadores dependientes de AWS Glue. Con los flujos de trabajo de AWS Glue, puede diseñar un proceso de ETL complejo de múltiples trabajos, múltiples rastreadores que AWS Glue puede ejecutar y rastrear como una sola entidad. Después de crear un flujo de trabajo y especificar los trabajos, rastreadores y desencadenadores del flujo de trabajo, puede ejecutar el flujo de trabajo bajo demanda o en función de una programación.

Temas

- Información general de los flujos de trabajo en AWS Glue
- Creación y desarrollo manual de un flujo de trabajo en AWS Glue

- Inicio de un flujo de trabajo de AWS Glue a partir de un evento de Amazon EventBridge
- Visualizar los eventos de EventBridge que iniciaron un flujo de trabajo
- Ejecución y supervisión de un flujo de trabajo en AWS Glue
- Detener una ejecución de flujo de trabajo
- Reparar y reanudar la ejecución de un flujo de trabajo
- Obtención y configuración de propiedades de ejecución de flujo de trabajo en AWS Glue
- Consulta de flujos de trabajo mediante la AWS Glue API
- Restricciones de esquemas y flujos de trabajo en AWS Glue
- Solución de errores de esquema en AWS Glue
- Permisos de personas y roles para esquemas de AWS Glue

Información general de los flujos de trabajo en AWS Glue

En AWS Glue, puede utilizar flujos de trabajo para crear y visualizar actividades de extracción, transformación y carga (ETL) complejas que implican varios rastreadores, trabajos y desencadenadores. Cada flujo de trabajo administra la ejecución y monitoreo de todos sus trabajos y rastreadores. Un flujo de trabajo registra el progreso de ejecución y el estado, ya que ejecuta cada componente. Esto le proporciona información general de la tarea de mayor envergadura y los detalles de cada paso. La consola de AWS Glue ofrece una representación visual de un flujo de trabajo en forma de gráfico.

Puede crear un flujo de trabajo desde un proyecto de AWS Glue, o puede crear manualmente un flujo de trabajo un componente a la vez mediante la herramienta AWS Management Console o AWS Glue API. Para obtener más información acerca de los esquemas, consulte the section called “Información general de los esquemas”.

Los desencadenadores dentro de los flujos de trabajo pueden habilitar los trabajos y rastreadores y pueden iniciarse cuando se completan los trabajos y rastreadores. Al usar desencadenadores, puede crear grandes cadenas de trabajos y rastreadores interdependientes. Además de los desencadenadores dentro de un flujo de trabajo que definen las dependencias de los trabajos y rastreadores, cada flujo de trabajo tiene un desencadenador de inicio. Existen tres tipos de desencadenadores de inicio:

- Programación: el flujo de trabajo se inicia según la programación que defina. La programación puede ser diaria, semanal, mensual, etc., o puede ser una programación personalizada basada en una expresión `cron`.

- Bajo demanda: el flujo de trabajo se inicia manualmente desde la consola de AWS Glue, la API o AWS CLI.
- Evento de EventBridge: el flujo de trabajo se inicia cuando se produce un único evento de Amazon EventBridge o un lote de eventos de Amazon EventBridge. Con este tipo de desencadenador, AWS Glue puede ser un consumidor de eventos en una arquitectura basada en eventos. Cualquier tipo de evento de EventBridge puede iniciar un flujo de trabajo. Un caso de uso común es la llegada de un nuevo objeto en un bucket de Amazon S3 (la operación `PutObject` de S3).

  Iniciar un flujo de trabajo con un lote de eventos significa esperar hasta que se haya recibido un número especificado de eventos o hasta que haya transcurrido un tiempo especificado. Al crear el desencadenador de eventos EventBridge, puede especificar condiciones de lote de forma opcional. Si especifica condiciones de lote, debe especificar el tamaño del lote (número de eventos) y puede especificar una ventana de lote (número de segundos) si lo desea. La ventana por lotes predeterminada y máxima es de 900 segundos (15 minutos). La condición del lote que se cumpla primero inicia el flujo de trabajo. La ventana del lote se inicia cuando llega el primer evento. Si no especifica las condiciones de lote al crear un desencadenador, el tamaño predeterminado del lote es 1.

  Cuando se inicia el flujo de trabajo, las condiciones del lote se restablecen y el desencadenador de eventos comienza a comprobar si se cumple la siguiente condición de lote para volver a iniciar el flujo de trabajo.

  En la siguiente tabla se muestra cómo el tamaño del lote y la ventana del lote funcionan juntos para habilitar un flujo de trabajo.

Tamaño de lote	Ventana del lote	Condición desencadenante resultante
10		El flujo de trabajo se desencadena ante la llegada de 10 eventos EventBridge, o 15 minutos después de la llegada del primer evento, lo que ocurra primero. (Si no se especifica el tamaño de ventana, el valor predeterm inado es de 15 minutos).
10	2 minutos	El flujo de trabajo se desencadena ante la llegada de 10 eventos EventBridge, o 2 minutos después de la llegada del primer evento, lo que ocurra primero.

Tamaño de lote	Ventana del lote	Condición desencadenante resultante
1		El flujo de trabajo se desencadena ante la llegada del primer evento. El tamaño de la ventana es irrelevante. Si no especifica las condiciones de lote al crear un desencadenador de evento EventBridge, el tamaño predeterminado del lote es 1.

La operación de la API `GetWorkflowRun` devuelve la condición de lote que desencadenó el flujo de trabajo.

Independientemente de cómo se inicie un flujo de trabajo, puede especificar el número máximo de ejecuciones concurrentes de flujo de trabajo al crear el flujo de trabajo.

Si un evento o lote de eventos inicia una ejecución de flujo de trabajo que eventualmente falla, ese evento o lote de eventos ya no se considera para iniciar una ejecución de flujo de trabajo. Una nueva ejecución de flujo de trabajo se inicia sólo cuando llega el siguiente evento o lote de eventos.

> **Important**
>
> Limite la cantidad total de trabajos, rastreadores y desencadenadores de un flujo de trabajo a 100 o menos. Si incluye más de 100, es posible que se produzcan errores al intentar reanudar o detener las ejecuciones del flujo de trabajo.

No se iniciará una ejecución de flujo de trabajo si excede el límite de concurrencia establecido para el flujo de trabajo, aunque se cumpla la condición del evento. Se aconseja ajustar los límites de concurrencia del flujo de trabajo en función del volumen de eventos esperado. AWS Glue no reintenta ejecutar los flujos de trabajo que presentan errores debido a límites de concurrencia excedidos. Del mismo modo, se aconseja ajustar los límites de concurrencia para trabajos y rastreadores dentro de flujos de trabajo en función del volumen de eventos esperado.

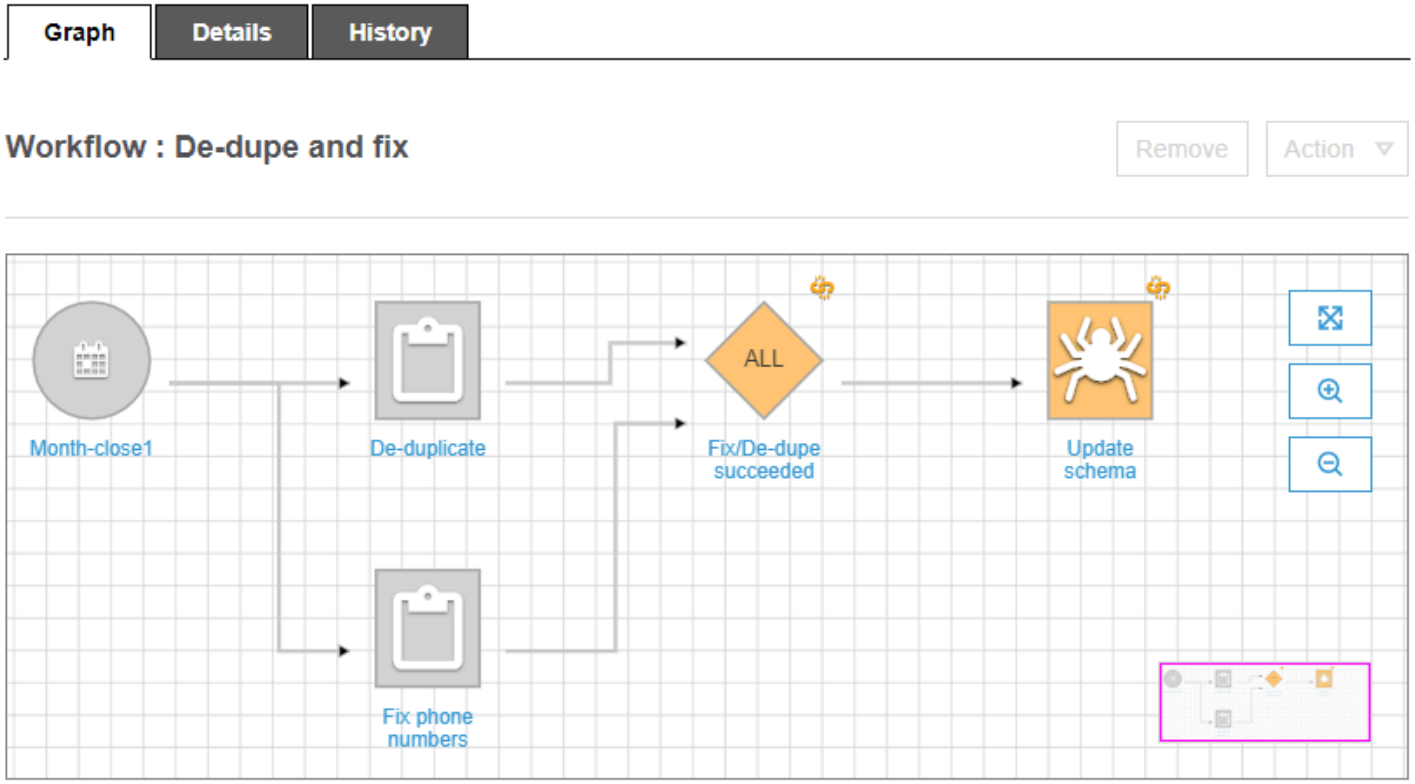
Propiedades de ejecución del flujo de trabajo

Para compartir y administrar el estado en toda la ejecución de flujo de trabajo, puede definir propiedades de ejecución de flujo de trabajo predeterminadas. Estas propiedades, que son pares nombre-valor, están disponibles para todos los trabajos en el flujo de trabajo. Utilice la AWS Glue API

para recuperar las propiedades de ejecución de flujo de trabajo y modifíquelas para los trabajos que vengan después en el flujo de trabajo.

Gráfico del flujo de trabajo

La imagen siguiente muestra el gráfico de un flujo de trabajo básico en la consola de AWS Glue. Su flujo de trabajo podría tener decenas de componentes.



Este flujo de trabajo se inicia mediante un desencadenador de programación, `Month-close1`, que inicia dos trabajos, `De-duplicate` y `Fix phone numbers`. Tras la realización completa de ambos trabajos, un desencadenador de eventos, `Fix/De-dupe succeeded`, inicia un rastreador, `Update schema`.

Vistas de flujo de trabajo estáticas y dinámicas

Para cada flujo de trabajo hay una noción de vista estática y dinámica. La vista estática indica el diseño del flujo de trabajo. La vista dinámica es una vista de tiempo de ejecución que incluye la información de ejecución más reciente para cada uno de los trabajos y rastreadores. La información de ejecución incluye detalles sobre el error y el estado de success (correcto).

Cuando un flujo de trabajo se encuentra en ejecución, la consola muestra la vista dinámica, que indica gráficamente los trabajos completados y que todavía se tienen que ejecutar. También puede

recuperar una vista dinámica de un flujo de trabajo en ejecución mediante la AWS Glue API. Para obtener más información, consulte Consulta de flujos de trabajo mediante la AWS Glue API.

Véase también

- the section called “Creación de un flujo de trabajo a partir de un esquema”
- the section called “Creación y desarrollo manual de un flujo de trabajo”
- Flujos de trabajo (para la API de flujos de trabajo)

Creación y desarrollo manual de un flujo de trabajo en AWS Glue

Puede usar la consola de AWS Glue para crear y construir un flujo de trabajo un nodo a la vez de forma manual.

Un flujo de trabajo contiene trabajos, rastreadores y desencadenadores. Antes de crear un flujo de trabajo de forma manual, cree los trabajos y rastreadores que el flujo de trabajo vaya a incluir. Es mejor especificar rastreadores de ejecución bajo demanda para los flujos de trabajo. Puede crear nuevos desencadenadores mientras se desarrolla su flujo de trabajo o puede clonar desencadenadores existentes en el flujo de trabajo. Al clonar un desencadenador, se agregan al flujo de trabajo todos los objetos del catálogo asociados al desencadenador (los trabajos o rastreadores que lo activan y los trabajos o rastreadores que lo inician).

Important

Limite la cantidad total de trabajos, rastreadores y desencadenadores de un flujo de trabajo a 100 o menos. Si incluye más de 100, es posible que se produzcan errores al intentar reanudar o detener las ejecuciones del flujo de trabajo.

Puede diseñar su propio flujo de trabajo añadiendo desencadenadores al gráfico de flujo de trabajo y definiendo los eventos vistos y las acciones para cada desencadenador. Comience con un desencadenador de arranque, que puede ser un desencadenador bajo demanda o programado, y complete el gráfico añadiendo desencadenadores de eventos (condicional).

Paso 1: Crear el flujo de trabajo

1. Inicie sesión en la AWS Management Console y abra la consola de AWS Glue en https://console.aws.amazon.com/glue/.
2. En el panel de navegación, en ETL, elija Workflows (Flujos de trabajo).
3. Seleccione Add workflow (Añadir flujo de trabajo) y complete el formulario Add a new ETL workflow (Añadir un nuevo flujo de trabajo de ETL).

   Las propiedades de ejecución predeterminadas opcionales que añada estarán disponibles como argumentos para todos los trabajos en el flujo de trabajo. Para obtener más información, consulte Obtención y configuración de propiedades de ejecución de flujo de trabajo en AWS Glue.
4. Seleccione Add workflow (Añadir flujo de trabajo).

   El nuevo flujo de trabajo aparecerá en la lista en la página Workflows (Flujos de trabajo).

Paso 2: Añadir un desencadenador de arranque

1. En la página Workflows (Flujos de trabajo), seleccione el nuevo flujo de trabajo. A continuación, en la parte inferior de la página, asegúrese de seleccionar la pestaña Graph (Gráfico).
2. Seleccione Add trigger (Añadir desencadenador), y en el cuadro de diálogo Add trigger (Añadir desencadenador), realice uno de los siguientes procedimientos:

   - Elija Clone existing (Clonación existente) y seleccione un desencadenador que clonar. A continuación, elija Add (Añadir).

     El desencadenador aparece en el gráfico, junto con los trabajos y los rastreadores que ve e inicia.

     Si ha seleccionado por error el desencadenador erróneo, seleccione el desencadenador en el gráfico y, a continuación, elija Remove (Eliminar).
   - Elija Add new (Añadir nuevo) y complete el formulario Add trigger (Añadir desencadenador).
     1. Para Trigger type (Tipo de desencadenador), seleccione Schedule (Programación), On demand (Bajo demanda), o bien EventBridge event (Evento de EventBridge).

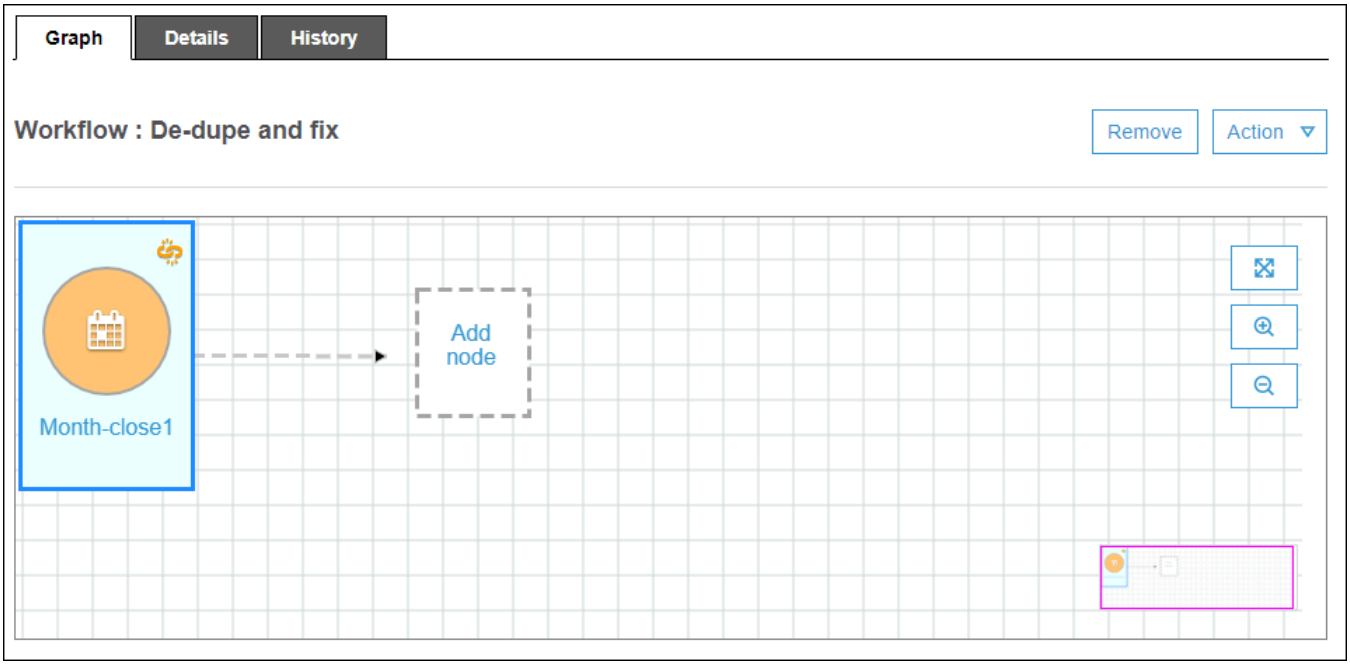
Para el tipo de desencadenador Schedule (Programación), elija una de las opciones en Frequency (Frecuencia). Seleccione Custom (Personalizado) e ingrese una expresión `cron`.

Para el tipo de desencadenador EventBridge, ingrese Number of events (Número de eventos) (tamaño del lote) y, opcionalmente, ingrese Time delay (Tiempo de retraso) (ventana por lotes). Si omite Time delay (Tiempo de retraso), la ventana por lotes tiene un valor predeterminado de 15 minutos. Para obtener más información, consulte Información general de los flujos de trabajo en AWS Glue.

2. Elija Add (Agregar).

El desencadenador aparece en el gráfico, junto con un nodo de marcador de posición (con la etiqueta Add node [Añadir nodo]). En el ejemplo a continuación, el desencadenador de inicio es un desencadenador de programación denominado `Month-close1`.

En este momento, el desencadenador no está guardado aún.



3. Si ha añadido un nuevo desencadenador, complete estos pasos:

   a. Realice una de las acciones siguientes:

      - Elija el nodo del marcador de posición (Add node [Añadir nodo]).

- Asegúrese de que el desencadenador de arranque esté seleccionado, y en el menú Action (Acción) de arriba del gráfico, elija Add jobs/crawlers to trigger (Añadir trabajos/rastreadores al desencadenador).

b. En el cuadro de diálogo Add jobs(s) and crawler(s) to trigger (Añadir trabajos y rastreadores al desencadenador), seleccione uno o más trabajos o desencadenadores y, a continuación, elija Add (Añadir).

   El desencadenador se guarda y los trabajos y rastreadores seleccionados aparecen en el gráfico con los conectores del desencadenador.

   Si ha añadido por error los trabajos y rastreadores erróneos, puede seleccionar el desencadenador o un conector y elegir Remove (Eliminar).

Paso 3: Agregar más desencadenadores

Continúe construyendo su flujo de trabajo al agregar más desencadenadores de tipo Event (Evento). Para ampliar o reducir la imagen, o para agrandar el lienzo del gráfico, utilice los iconos a la derecha del gráfico. Para agregar los desencadenadores, complete los siguientes pasos:

> **Note**
>
> No hay ninguna acción para guardar el flujo de trabajo. Después de agregar el último desencadenador y asignar acciones al desencadenador, el flujo de trabajo se completa y se guarda. Siempre puede volver más tarde y agregar más nodos.

1. Realice una de las acciones siguientes:

   - Para clonar un desencadenador existente, asegúrese de que no se selecciona ningún nodo en el gráfico y, en el menú Action (Acción), elija Add trigger (Añadir desencadenador).
   - Para añadir un nuevo desencadenador que vea un trabajo o rastreador determinado en el gráfico, seleccione el nodo de trabajo o rastreador y, a continuación, elija el nodo de marcador de posición Add trigger (Añadir desencadenador).

     Puede añadir más trabajos o rastreadores para ver este desencadenador en un paso posterior.

2. En el cuadro de diálogo Add trigger (Añadir desencadenador), realice una de las acciones siguientes:

- Elija Add new (Añadir nuevo) y complete el formulario Add trigger (Añadir desencadenador). A continuación, elija Add (Añadir).

  El desencadenador aparece en el gráfico. Se completará el desencadenador en un paso posterior.

- Elija Clone existing (Clonación existente) y seleccione un desencadenador que clonar. A continuación, elija Add (Añadir).

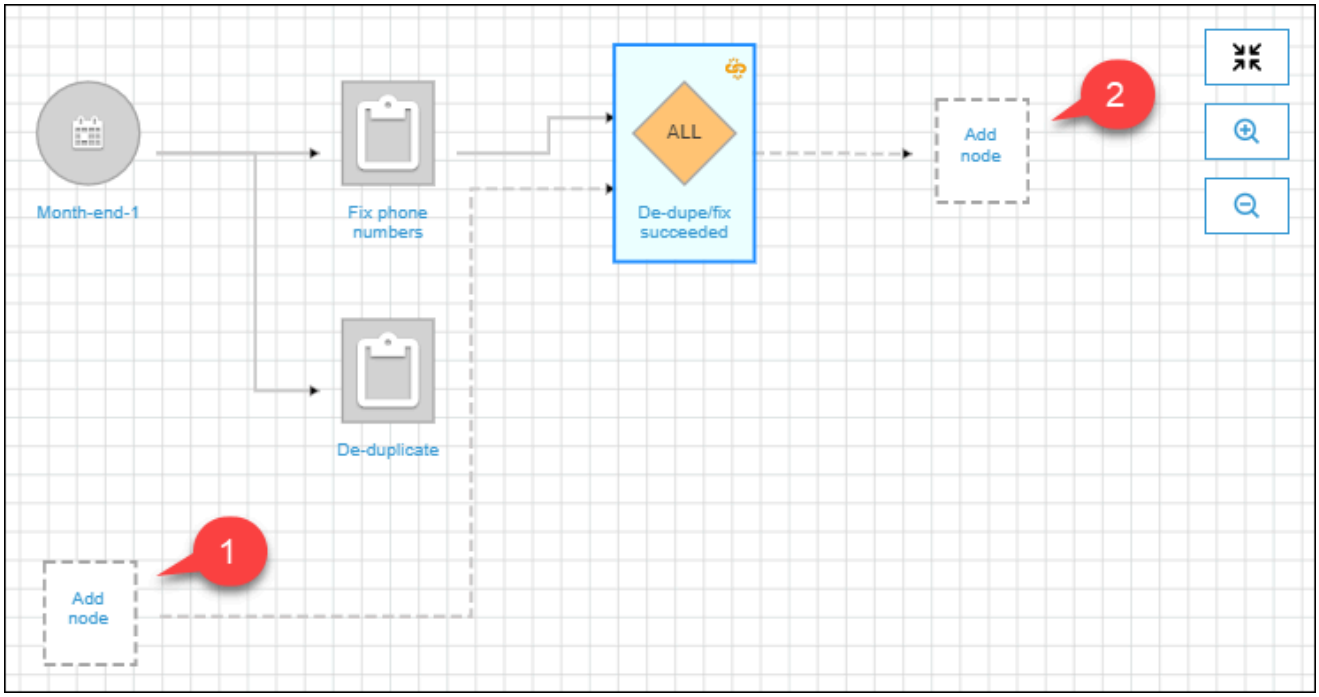
  El desencadenador aparece en el gráfico, junto con los trabajos y los rastreadores que ve e inicia.

  Si ha elegido por error el desencadenador erróneo, seleccione el desencadenador en el gráfico y, a continuación, elija Remove (Eliminar).

3. Si ha añadido un nuevo desencadenador, complete estos pasos:

   a. Seleccione el nuevo desencadenador.

      Como se muestra en el siguiente gráfico, se selecciona el desencadenador `De-dupe/fix succeeded`, y los nodos de marcador de posición aparecen para (1) eventos para ver y (2) acciones.



   b. (Opcional si el desencadenador ya ve un evento y desea añadir más trabajos o rastreadores que ver). Elija el nodo de marcador de posición de eventos que ver y, en el cuadro de

diálogo Add job(s) and crawler(s) to watch (Añadir trabajos y rastreadores que ver), seleccione uno o más trabajos o rastreadores. Elija un evento que ver (SUCCEEDED (Correcto), FAILED (Erróneo), etc.), y elija Add (Añadir).

c. Asegúrese de que se seleccione el desencadenador y elija el nodo de marcador de posición de acciones.

d. En el cuadro de diálogo Add jobs(s) and crawler(s) to watch (Añadir trabajos y rastreadores que ver), seleccione uno o más trabajos o desencadenadores y, a continuación, elija Add (Añadir).

   Los trabajos y rastreadores seleccionados aparecen en el gráfico con los conectores del desencadenador.

Para obtener más información sobre flujos de trabajo y esquemas, consulte los siguientes temas.

- Información general de los flujos de trabajo en AWS Glue
- Ejecución y supervisión de un flujo de trabajo en AWS Glue
- Creación de un flujo de trabajo a partir de un esquema en AWS Glue

Inicio de un flujo de trabajo de AWS Glue a partir de un evento de Amazon EventBridge

Amazon EventBridge, también conocido como CloudWatch Events, le permite automatizar los servicios de AWS y responder automáticamente a eventos del sistema, como problemas de disponibilidad de aplicaciones o cambios de recursos. Los eventos de los servicios de AWS se envían a EventBridge casi en tiempo real. Puede crear reglas sencillas para indicar qué eventos le resultan de interés, así como qué acciones automatizadas se van a realizar cuando un evento cumple una de las reglas.

Con el soporte de EventBridge, AWS Glue puede funcionar como productor y consumidor de eventos en una arquitectura basada en eventos. Para los flujos de trabajo, AWS Glue soporta cualquier tipo de evento de EventBridge como consumidor. Es probable que el caso de uso más común sea la llegada de un nuevo objeto en un bucket de Amazon S3. Si tiene datos que llegan a intervalos irregulares o indefinidos, puede procesar estos datos lo más rápido posible luego de su llegada.

> **Note**
>
> AWS Glue no garantiza la entrega de mensajes de EventBridge. AWS Glue no realiza desduplicación si EventBridge entrega mensajes duplicados. Debe administrar la idempotencia en función del caso de uso.
> Asegúrese de configurar las reglas de EventBridge en forma correcta para evitar el envío de eventos no deseados.

Antes de empezar

Si desea iniciar un flujo de trabajo con eventos de datos de Amazon S3, debe asegurarse de que los eventos del bucket de S3 de interés se registren en AWS CloudTrail y EventBridge. Para ello, debe crear un seguimiento de CloudTrail. Para obtener más información, consulte Crear un seguimiento para su cuenta de AWS.

Para iniciar un flujo de trabajo con un evento de EventBridge

> **Note**
>
> En los siguientes comandos, sustituya:
>
> - *<workflow-name>* con el nombre que se va a asignar al flujo de trabajo.
> - *<trigger-name>* con el nombre que se va a asignar al desencadenador.
> - *<bucket-name>* con el nombre del bucket de Amazon S3.
> - *<account-id>* con un ID de cuenta de AWS válido.
> - *<region>* con el nombre de la región (por ejemplo, `us-east-1`).
> - *<rule-name>* con el nombre que se asignará a la regla de EventBridge.

1. Asegúrese de tener permisos de AWS Identity and Access Management (IAM) para crear y ver reglas y destinos de EventBridge. A continuación, se muestra una política de ejemplo que puede asociar. Es posible que desee reducir el alcance para poner límites a las operaciones y los recursos.

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Effect": "Allow",
            "Action": [
                "events:PutRule",
                "events:DisableRule",
                "events:DeleteRule",
                "events:PutTargets",
                "events:RemoveTargets",
                "events:EnableRule",
                "events:List*",
                "events:Describe*"
            ],
            "Resource": "*"
        }
    ]
}
```

2. Cree un rol de IAM que el servicio EventBridge pueda asumir al enviar un evento a AWS Glue.

   a. En la página Create role (Crear rol) de la consola de IAM, elija AWS Service (Servicio de AWS). A continuación, elija el servicio CloudWatch Events (Eventos de CloudWatch).

   b. Finalice el asistente Create role (Creación de rol). El asistente adjunta automáticamente las políticas `CloudWatchEventsBuiltInTargetExecutionAccess` y `CloudWatchEventsInvocationAccess`.

   c. Asocie la siguiente política en línea al rol. Esta política permite que el servicio EventBridge dirija eventos a AWS Glue.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "glue:notifyEvent"
      ],
      "Resource": [
        "arn:aws:glue:<region>:<account-id>:workflow/<workflow-name>"
      ]
    }
  ]
}
```

3. Ingrese el siguiente comando para crear el flujo de trabajo.

   Consulte crear flujo de trabajo en la Referencia de comandos de AWS CLI para obtener información acerca de los parámetros adicionales de línea de comandos opcionales.

```
aws glue create-workflow --name <workflow-name>
```

4. Ingrese el siguiente comando para crear un desencadenador de eventos de EventBridge para el flujo de trabajo. Este será el desencadenador de inicio del flujo de trabajo. Reemplace *<actions>* con las acciones a realizar (los trabajos y los rastreadores de inicio).

   Consulte crear un desencadenador en la Referencia de comandos de AWS CLI para obtener información acerca de cómo codificar el argumento de `actions`.

```
aws glue create-trigger --workflow-name <workflow-name> --type EVENT --
name <trigger-name> --actions <actions>
```

   Si desea que el flujo de trabajo se desencadene mediante un lote de eventos en lugar de un solo evento EventBridge, ingrese el siguiente comando en su lugar.

```
aws glue create-trigger --workflow-name <workflow-name> --type EVENT
 --name <trigger-name> --event-batching-condition BatchSize=<number-of-
events>,BatchWindow=<seconds> --actions <actions>
```

   Para el argumento de `event-batching-condition`, se requiere el `BatchSize`, mientras que la `BatchWindow` es opcional. Si se omite `BatchWindow`, el valor predeterminado de la ventana es 900 segundos, que es el tamaño máximo de la ventana.

   Example

   En el ejemplo siguiente se crea un desencadenador que inicia el flujo de trabajo de `eventtest`, después de que hayan llegado tres eventos de EventBridge, o cinco minutos después de que llegue el primer evento, lo que ocurra primero.

```
aws glue create-trigger --workflow-name eventtest --type EVENT --name objectArrival
 --event-batching-condition BatchSize=3,BatchWindow=300 --actions JobName=test1
```

5. Cree una regla en Amazon EventBridge.

   a. Cree el objeto JSON para los detalles de la regla en el editor de texto que prefiera.

El ejemplo a continuación indica que Amazon S3 es el origen del evento, `PutObject` es el nombre del evento y el nombre del bucket aparece como parámetro de solicitud. Esta regla inicia un flujo de trabajo cuando llega un nuevo objeto al bucket.

```
{
  "source": [
    "aws.s3"
  ],
  "detail-type": [
    "AWS API Call via CloudTrail"
  ],
  "detail": {
    "eventSource": [
      "s3.amazonaws.com"
    ],
    "eventName": [
      "PutObject"
    ],
    "requestParameters": {
      "bucketName": [
        "<bucket-name>"
      ]
    }
  }
}
```

Para iniciar el flujo de trabajo cuando un nuevo objeto llega a una carpeta dentro del bucket, puede sustituir el código siguiente por `requestParameters`.

```
    "requestParameters": {
      "bucketName": [
        "<bucket-name>"
      ]
      "key" : [{ "prefix" : "<folder1>/<folder2>/*"}}]
  }
```

b. Utilice su herramienta preferida para convertir el objeto JSON de regla en una cadena de escape.

```
{\n  \"source\": [\n    \"aws.s3\"\n  ],\n  \"detail-type\": [\n    \"AWS API
 Call via CloudTrail\"\n  ],\n  \"detail\": {\n    \"eventSource\": [\n
```

```
 \"s3.amazonaws.com\"\n    ],\n    \"eventName\": [\n      \"PutObject\"\n
  ],\n    \"requestParameters\": {\n      \"bucketName\": [\n        \"<bucket-
name>\"\n      ]\n    }\n  }\n}
```

c. Ejecute el siguiente comando para crear una plantilla de parámetros JSON que pueda editar para especificar los parámetros de entrada a un comando de `put-rule` posterior. Guarde el resultado en un archivo. En este ejemplo, el archivo se llama `ruleCommand`.

```
aws events put-rule --name <rule-name> --generate-cli-skeleton >ruleCommand
```

Para obtener más información sobre el parámetro `--generate-cli-skeleton`, consulte Generar esqueleto y parámetros de entrada de AWS CLI a partir de un archivo de entrada JSON o YAML en la Guía del usuario de la interfaz de línea de comandos de AWS.

El archivo resultante debe tener el siguiente aspecto.

```
{
    "Name": "",
    "ScheduleExpression": "",
    "EventPattern": "",
    "State": "ENABLED",
    "Description": "",
    "RoleArn": "",
    "Tags": [
        {
            "Key": "",
            "Value": ""
        }
    ],
    "EventBusName": ""
}
```

d. Edite el archivo para eliminar parámetros de manera opcional y para especificar como mínimo los parámetros de `Name`, `EventPattern` y `State`. Para el parámetro `EventPattern`, proporcione la cadena de escape para los detalles de la regla que creó en un paso anterior.

```
{
    "Name": "<rule-name>",
    "EventPattern": "{\n  \"source\": [\n    \"aws.s3\"\n  ],\n  \"detail-
type\": [\n    \"AWS API Call via CloudTrail\"\n  ],\n  \"detail\": {\n
```

```
 \"eventSource\": [\n      \"s3.amazonaws.com\"\n    ],\n    \"eventName\": [\n
      \"PutObject\"\n    ],\n    \"requestParameters\": {\n      \"bucketName
\": [\n        \"<bucket-name>\"\n      ]\n    }\n  }\n}",
    "State": "DISABLED",
    "Description": "Start an AWS Glue workflow upon new file arrival in an
 Amazon S3 bucket"
}
```

> **Note**
>
> Es mejor dejar la regla deshabilitada hasta que termine de crear el flujo de trabajo.

e. Ingrese el siguiente comando `put-rule`, que lee los parámetros de entrada del `ruleCommand` del archivo.

```
aws events put-rule --name <rule-name> --cli-input-json file://ruleCommand
```

El siguiente resultado indica éxito.

```
{
    "RuleArn": "<rule-arn>"
}
```

6. Ingrese el siguiente comando para adjuntar la regla a un destino. El destino es el flujo de trabajo en AWS Glue. Reemplace *<role-name>* con el rol que ha creado al principio de este procedimiento.

```
aws events put-targets --rule <rule-name> --targets
 "Id"="1","Arn"="arn:aws:glue:<region>:<account-id>:workflow/<workflow-
name>","RoleArn"="arn:aws:iam::<account-id>:role/<role-name>" --region <region>
```

El siguiente resultado indica éxito.

```
{
    "FailedEntryCount": 0,
    "FailedEntries": []
}
```

7. Confirme la conexión correcta de la regla y el destino al introducir el siguiente comando.

```
aws events list-rule-names-by-target --target-arn arn:aws:glue:<region>:<account-id>:workflow/<workflow-name>
```

El siguiente resultado indica el éxito, donde *<rule-name>* es el nombre de la regla que creó.

```
{
    "RuleNames": [
        "<rule-name>"
    ]
}
```

8. Inicie sesión en la AWS Management Console y abra la consola de AWS Glue en https://console.aws.amazon.com/glue/.
9. Seleccione el flujo de trabajo y compruebe que el desencadenador de inicio y sus acciones (los trabajos o rastreadores que inicia) aparezcan en el gráfico del flujo de trabajo. Luego, continúe con el procedimiento en Paso 3: Agregar más desencadenadores. O agregue más componentes al flujo de trabajo mediante la API de AWS Glue o AWS Command Line Interface.
10. Cuando el flujo de trabajo esté completamente especificado, habilite la regla.

```
aws events enable-rule --name <rule-name>
```

El flujo de trabajo está ahora listo para ser iniciado por un evento de EventBridge o un lote de eventos.

Véase también

- Guía del usuario de Amazon EventBridge
- Información general de los flujos de trabajo en AWS Glue
- Creación y desarrollo manual de un flujo de trabajo en AWS Glue

Visualizar los eventos de EventBridge que iniciaron un flujo de trabajo

Puede ver el ID del evento de Amazon EventBridge que inició el flujo de trabajo. Si el flujo de trabajo se inició mediante un lote de eventos, puede ver los ID de eventos de todos los eventos del lote.

Para flujos de trabajo con un tamaño de lote mayor que uno, también puede ver qué condición de lote inició el flujo de trabajo: la llegada del número de eventos en el tamaño del lote o el vencimiento de la ventana del lote.

Para visualizar los eventos de EventBridge que iniciaron un flujo de trabajo (consola)

1. Inicie sesión en la AWS Management Console y abra la consola de AWS Glue en https://console.aws.amazon.com/glue/.
2. En el panel de navegación, elija Workflows (Flujos de trabajo).
3. Seleccione el flujo de trabajo. Luego, en la parte inferior, elija la pestaña History (Historial).
4. Seleccione una ejecución de flujo de trabajo y, a continuación, elija View run details (Visualizar detalles de la ejecución).
5. En la página de detalles de la ejecución, localice el campo Run properties (Propiedades de la ejecución) y busque la clave was:eventIds.

   El valor de esa clave es una lista de ID de eventos de EventBridge.

Para visualizar los eventos de EventBridge que iniciaron un flujo de trabajo (API de AWS)

- Incluya el siguiente código en su script de Python.

```
workflow_params =
 glue_client.get_workflow_run_properties(Name=workflow_name,RunId=workflow_run_id)
batched_events = workflow_params['aws:eventIds']
```

  `batched_events` será una lista de cadenas, donde cada cadena es un ID de evento.

> Véase también
>
> - Guía del usuario de Amazon EventBridge
> - the section called “Descripción general de flujos de trabajo”

Ejecución y supervisión de un flujo de trabajo en AWS Glue

Si el desencadenador de inicio de un flujo de trabajo es un desencadenador bajo demanda, puede iniciar el flujo de trabajo desde la consola de AWS Glue. Siga los pasos a continuación para ejecutar y monitorear un flujo de trabajo. Si se produce un error en el flujo de trabajo, puede ver el gráfico de ejecución para determinar el nodo fallido. Para solucionar problemas, si el flujo de trabajo se creó a partir de un proyecto, puede visualizar la ejecución del proyecto para conocer los valores de parámetros del proyecto que se utilizaron para crear el flujo de trabajo. Para obtener más información, consulte the section called “Visualización de ejecuciones de esquemas”.

Puede ejecutar y monitorear un flujo de trabajo mediante la consola de AWS Glue, la API o AWS Command Line Interface (AWS CLI).

Para ejecutar y monitorear un flujo de trabajo (consola)

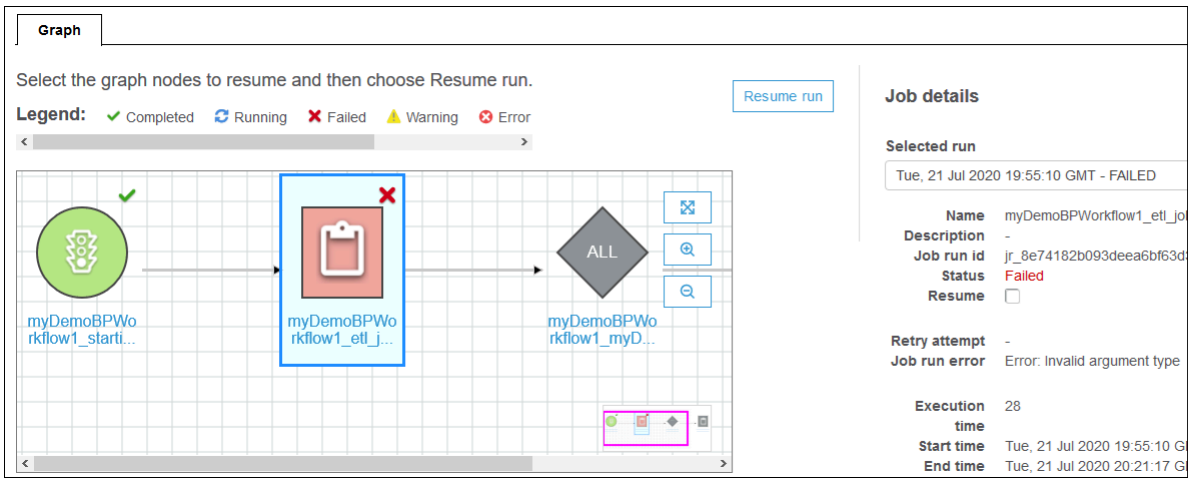
1. Abra la consola de AWS Glue en https://console.aws.amazon.com/glue/.
2. En el panel de navegación, en ETL, elija Workflows (Flujos de trabajo).
3. Seleccione un flujo de trabajo. En el menú Actions (Acciones), elija Run (Ejecutar).
4. Compruebe la columna Last run status (Estado de la última ejecución) en la lista de flujos de trabajo. Elija el botón de actualización para ver el estado del flujo de trabajo en curso.
5. Mientras el flujo de trabajo se está ejecutando o después de que se haya completado (o haya fallado), vea los detalles de ejecución a través de los pasos siguientes.

    a. Asegúrese de que se seleccione el flujo de trabajo y elija la pestaña History (Historial).

    b. Elija la ejecución del flujo de trabajo actual o más reciente y, a continuación, elija View run details (Ver detalles de la ejecución).

    El gráfico de tiempo de ejecución del flujo de trabajo muestra el estado actual de ejecución.

    c. Elija cualquier nodo del gráfico para ver los detalles y el estado del nodo.



Para ejecutar y monitorear un flujo de trabajo (AWS CLI)

1. Escriba el siguiente comando. Reemplace *<workflow-name>* con el flujo de trabajo que se va a ejecutar.

```
aws glue start-workflow-run --name <workflow-name>
```

   Si el flujo de trabajo se inicia correctamente, el comando devuelve el ID de ejecución.

2. Visualice el estado de ejecución del flujo de trabajo mediante el comando `get-workflow-run`. Proporcione el nombre e ID de ejecución del flujo de trabajo.

```
aws glue get-workflow-run --name myWorkflow --run-id
 wr_d2af14217e8eae775ba7b1fc6fc7a42c795aed3cbcd8763f9415452e2dbc8705
```

   A continuación, se muestra un ejemplo del resultado del comando.

```
{
    "Run": {
        "Name": "myWorkflow",
        "WorkflowRunId":
 "wr_d2af14217e8eae775ba7b1fc6fc7a42c795aed3cbcd8763f9415452e2dbc8705",
        "WorkflowRunProperties": {
            "run_state": "COMPLETED",
            "unique_id": "fee63f30-c512-4742-a9b1-7c8183bdaae2"
        },
        "StartedOn": 1578556843.049,
        "CompletedOn": 1578558649.928,
```

```
        "Status": "COMPLETED",
        "Statistics": {
            "TotalActions": 11,
            "TimeoutActions": 0,
            "FailedActions": 0,
            "StoppedActions": 0,
            "SucceededActions": 9,
            "RunningActions": 0,
            "ErroredActions": 0
        }
    }
}
```

Véase también:

- the section called “Descripción general de flujos de trabajo”
- the section called “Información general de los esquemas”

Detener una ejecución de flujo de trabajo

Puede usar la consola de AWS Glue, AWS Command Line Interface (AWS CLI) o la API de AWS Glue para detener una ejecución de flujo de trabajo. Cuando detiene una ejecución de flujo de trabajo, todos los trabajos y rastreadores en ejecución finalizan inmediatamente y los trabajos y rastreadores que aún no se han iniciado nunca se llegan a iniciar. Puede tardar hasta un minuto para que todos los trabajos en ejecución y rastreadores se detengan. El estado de ejecución del flujo de trabajo pasa de Running (Ejecución) a Stopping (Detención) y cuando la ejecución del flujo de trabajo está completamente detenida, el estado pasa a Stopped (Detenido).

Después de detener la ejecución del flujo de trabajo, puede ver el gráfico de ejecución para ver qué trabajos y rastreadores se han completado y cuáles nunca se han iniciado. A continuación, puede determinar si debe realizar algún paso para garantizar la integridad de los datos. Detener una ejecución de flujo de trabajo hace que no se realicen operaciones de restauración automática.

Para detener una ejecución de flujo de trabajo (consola)

1. Abra la consola de AWS Glue en https://console.aws.amazon.com/glue/.
2. En el panel de navegación, en ETL, elija Workflows (Flujos de trabajo).

3. Elija un flujo de trabajo en ejecución y, a continuación, elija la pestaña History (Historial).
4. Elija la ejecución del flujo de trabajo y, a continuación, elija Stop run (Detener ejecución).

   El estado de ejecución cambia a Stopping (Detención).
5. (Opcional) seleccione la ejecución del flujo de trabajo, elija View run details (Ver detalles de la ejecución) y revise el gráfico de ejecución.

Para detener una ejecución de flujo de trabajo (AWS CLI)

- Escriba el siguiente comando. Reemplace *<workflow-name>* por el nombre del flujo de trabajo y *<run-id>* por el ID de ejecución del flujo de trabajo que se va a detener.

```
aws glue stop-workflow-run --name <workflow-name> --run-id <run-id>
```

  A continuación, se muestra un ejemplo del comando stop-workflow-run.

```
aws glue stop-workflow-run --name my-workflow --run-id
 wr_137b88917411d128081069901e4a80595d97f719282094b7f271d09576770354
```

Reparar y reanudar la ejecución de un flujo de trabajo

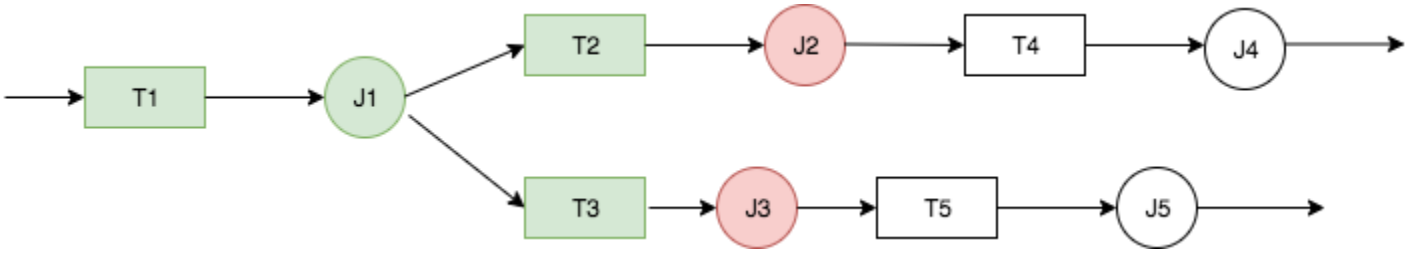
Si uno o más nodos (trabajos o rastreadores) de un flujo de trabajo no se completan en forma correcta, esto significa que el flujo de trabajo sólo se ejecutó en forma parcial. Después de encontrar las causas raíz y realizar correcciones, puede seleccionar uno o varios nodos a partir de los cuales reanudará la ejecución del flujo de trabajo y, a continuación, retomar la ejecución del flujo de trabajo. Se ejecutarán los nodos seleccionados y todos los nodos descendentes.

Temas

- Reanudar la ejecución de un flujo de trabajo: funcionamiento
- Reanudar una ejecución de flujo de trabajo
- Notas y limitaciones para reanudar las ejecuciones de flujos de trabajo

Reanudar la ejecución de un flujo de trabajo: funcionamiento

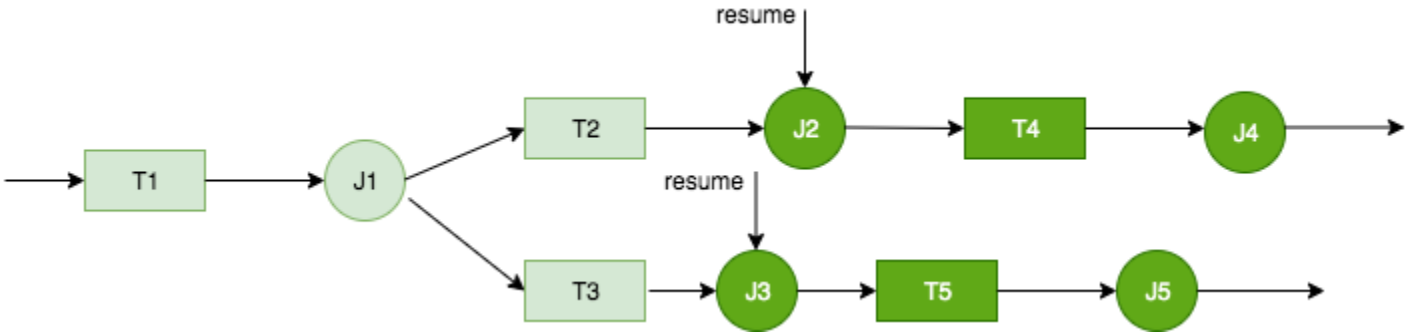
Considere el flujo de trabajo W1 en el siguiente diagrama.



La ejecución del flujo de trabajo continúa de la siguiente manera:

1. El desencadenador T1 inicia el trabajo J1.
2. La finalización exitosa de J1 activa los desencadenadores T2 y T3, que ejecutan los trabajos J2 y J3, respectivamente.
3. Los trabajos J2 y J3 fallan.
4. Los desencadenadores T4 y T5 no se activan ya que dependen de la finalización exitosa de J2 y J3, por lo tanto, los trabajos J4 y J5 no se ejecutan. El flujo de trabajo W1 sólo se ejecuta en forma parcial.

Suponga que se corrigen los problemas que causaron que J2 y J3 fallaran. J2 y J3 se seleccionan como puntos de partida para reanudar la ejecución del flujo de trabajo.



La ejecución del flujo de trabajo se reanuda de la siguiente manera:

1. Los trabajos J2 y J3 se ejecutan con éxito.
2. Se activan los desencadenadores T4 y T5.
3. Los trabajos J4 y J5 se ejecutan con éxito.

La ejecución del flujo de trabajo reanudado se realiza como una ejecución de flujo de trabajo independiente con un nuevo ID de ejecución. Cuando visualice el historial del flujo de trabajo, puede ver el ID de ejecución anterior para cualquier ejecución de flujo de trabajo. En el ejemplo de la

siguiente captura de pantalla, la ejecución del flujo de trabajo con el ID de ejecución `wr_c7a22...` (segunda fila) tenía un nodo que no se completó. El usuario solucionó el problema y reanudó la ejecución del flujo de trabajo, lo que dio lugar a un ID de ejecución `wr_a07e55...` (primera fila).

> **Note**
>
> De aquí en adelante, el término “ejecución de flujo de trabajo reanudado” hace referencia a la ejecución del flujo de trabajo que se creó cuando se reanudó la ejecución del flujo de trabajo anterior. La “ejecución de flujo de trabajo original” se refiere a la ejecución del flujo de trabajo que sólo se ejecutó en forma parcial y que debió reanudarse.

Gráfico de ejecución de flujo de trabajo reanudado

En una ejecución de flujo de trabajo reanudado, aunque sólo se ejecuta un subconjunto de nodos, el gráfico de ejecución es un gráfico completo. Es decir, los nodos que no se ejecutaron en el flujo de trabajo reanudado se copian del gráfico de ejecución de flujo de trabajo original. Los nodos de trabajo y rastreador copiados que se ejecutaron en la ejecución de flujo de trabajo original incluyen detalles de la ejecución.

Considere una vez más el flujo de trabajo W1 en el diagrama anterior. Cuando se reanuda la ejecución del flujo de trabajo a partir de J2 y J3, el gráfico de ejecución de flujo de trabajo reanudado muestra todos los trabajos, de J1 a J5, y todos los desencadenadores, de T1 a T5. Los detalles de la ejecución de J1 se copian desde la ejecución de flujo de trabajo original.

Instantáneas de flujos de trabajo

Cuando se inicia una ejecución de flujo de trabajo, AWS Glue toma una instantánea del gráfico de diseño del flujo de trabajo en ese momento. Esa instantánea se utiliza durante toda la ejecución del flujo de trabajo. Si realiza cambios en los desencadenadores después de que se inicie la ejecución, esos cambios no afectan la ejecución del flujo de trabajo actual. Las instantáneas garantizan que las ejecuciones de flujo de trabajo se realicen de manera coherente.

Las instantáneas hacen que solo los desencadenadores sean inmutables. Los cambios que realice en los trabajos y rastreadores descendentes durante la ejecución del flujo de trabajo surtirán efecto para la ejecución actual.

Reanudar una ejecución de flujo de trabajo

Siga estos pasos para reanudar una ejecución de flujo de trabajo. Puede reanudar una ejecución de flujo de trabajo mediante la consola de AWS Glue, la API, o AWS Command Line Interface (AWS CLI).

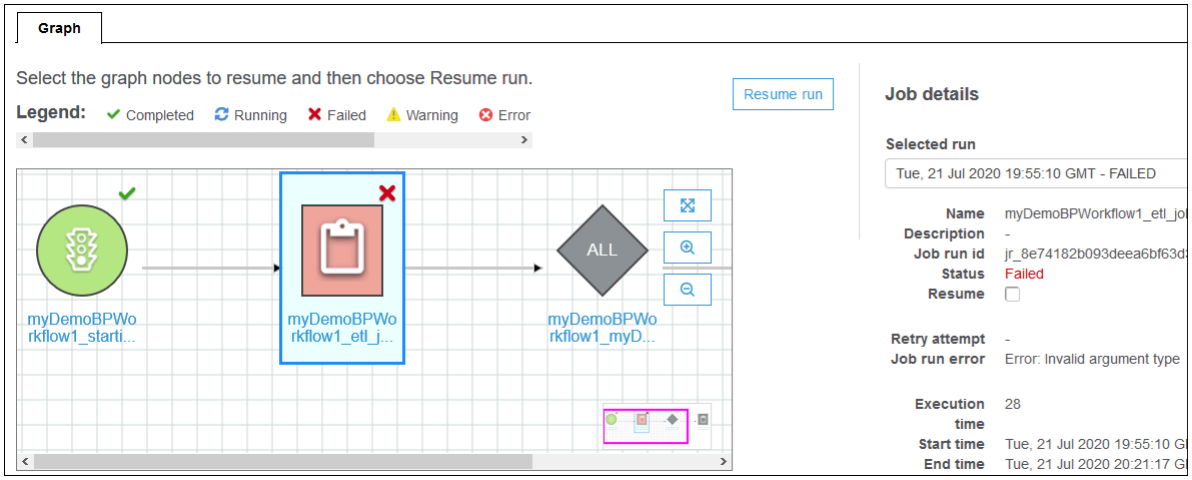
Para reanudar un flujo de trabajo (consola)

1. Abra la consola de AWS Glue en https://console.aws.amazon.com/glue/.

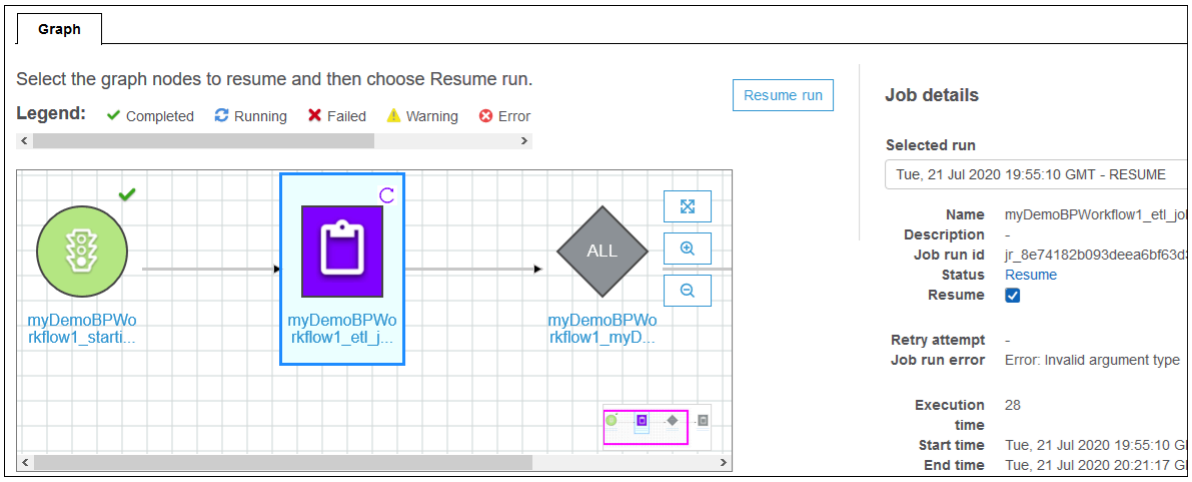
   Inicie sesión como usuario con permisos para visualizar flujos de trabajo y reanudar ejecuciones de flujos de trabajo.

   > **Note**
   >
   > Para reanudar ejecuciones de flujo de trabajo, necesita el permiso `glue:ResumeWorkflowRun` de AWS Identity and Access Management (IAM).

2. En el panel de navegación, elija Workflows (Flujos de trabajo).
3. Seleccione un flujo de trabajo y, a continuación, elija la pestaña History (Historial).
4. Seleccione la ejecución de flujo de trabajo que se ejecutó sólo en forma parcial y, a continuación, elija View run details (Visualizar los detalles de la ejecución).
5. En el gráfico de ejecución, seleccione el primer (o único) nodo que desea reiniciar y desde el que desea reanudar la ejecución del flujo de trabajo.
6. En el panel de detalles situado a la derecha del gráfico, seleccione la casilla de verificación Resume (Reanudar).



El nodo cambia de color y muestra un pequeño ícono de reanudación en la parte superior derecha.



7. Complete los dos pasos anteriores para reiniciar cualquier otro nodo adicional.
8. Elija Resume run (Reanudar ejecución).

Para retomar la ejecución de un flujo de trabajo (AWS CLI)

1. Asegúrese de que dispone del permiso `glue:ResumeWorkflowRun` de IAM.
2. Recupere los ID de nodo para los nodos que desee reiniciar.

   a. Ejecute el comando `get-workflow-run` para la ejecución del flujo de trabajo original. Proporcione el nombre del flujo de trabajo y el ID de ejecución, y agregue la opción `--include-graph`, como se muestra en el ejemplo a continuación. Obtenga el ID de ejecución de la pestaña History (Historial) en la consola, o ejecute el comando `get-workflow`.

```
aws glue get-workflow-run --name cloudtrailtest1 --run-id
 wr_a07e55f2087afdd415a404403f644a4265278f68b13ba3da08c71924ebe3c3a8 --include-
graph
```

El comando devuelve los nodos y los bordes del gráfico como un objeto JSON grande.

b. Localice los nodos de interés en función de las propiedades de `Type` y `Name` de los objetos del nodo.

A continuación, se muestra un resultado de objeto de nodo a modo de ejemplo.

```
{
    "Type": "JOB",
    "Name": "test1_post_failure_4592978",
    "UniqueId":
 "wnode_d1b2563c503078b153142ee76ce545fe5ceef66e053628a786ddd74a05da86fd",
    "JobDetails": {
        "JobRuns": [
            {
                "Id":
 "jr_690b9f7fc5cb399204bc542c6c956f39934496a5d665a42de891e5b01f59e613",
                "Attempt": 0,
                "TriggerName": "test1_aggregate_failure_649b2432",
                "JobName": "test1_post_failure_4592978",
                "StartedOn": 1595358275.375,
                "LastModifiedOn": 1595358298.785,
                "CompletedOn": 1595358298.785,
                "JobRunState": "FAILED",
                "PredecessorRuns": [],
                "AllocatedCapacity": 0,
                "ExecutionTime": 16,
                "Timeout": 2880,
                "MaxCapacity": 0.0625,
                "LogGroupName": "/aws-glue/python-jobs"
            }
        ]
    }
}
```

c. Obtenga el ID del nodo de la propiedad `UniqueId` del objeto del nodo.

3. Ejecute el comando `resume-workflow-run`. Especifique el nombre del flujo de trabajo, el ID de ejecución y la lista de ID de nodos separados por espacios, como se muestra en el siguiente ejemplo.

```
aws glue resume-workflow-run --name cloudtrailtest1 --run-id
 wr_a07e55f2087afdd415a404403f644a4265278f68b13ba3da08c71924ebe3c3a8 --node-
ids wnode_ca1f63e918fb855e063aed2f42ec5762ccf71b80082ae2eb5daeb8052442f2f3
 wnode_d1b2563c503078b153142ee76ce545fe5ceef66e053628a786ddd74a05da86fd
```

El comando arroja el ID de ejecución de la ejecución del flujo de trabajo reanudado (nuevo) y una lista de los nodos que se iniciarán.

```
{
    "RunId": "wr_2ada0d3209a262fc1156e4291134b3bd643491bcfb0ceead30bd3e4efac24de9",
    "NodeIds": [
        "wnode_ca1f63e918fb855e063aed2f42ec5762ccf71b80082ae2eb5daeb8052442f2f3"
    ]
}
```

Tenga en cuenta que aunque el comando `resume-workflow-run` de ejemplo enumeró dos nodos que se reiniciarán, el resultado de ejemplo indicaba que sólo se reiniciaría un nodo. Esto se debe a que un nodo era descendente al otro, y el nodo descendente se reiniciaría de todos modos como resultado de los pasos habituales del flujo de trabajo.

Notas y limitaciones para reanudar las ejecuciones de flujos de trabajo

Tenga en cuenta las siguientes notas y limitaciones al reanudar las ejecuciones de flujos de trabajo.

- Puede reanudar la ejecución de un flujo de trabajo sólo si se encuentra en el estado `COMPLETED`.

> **Note**
>
> Incluso si uno o más nodos en la ejecución de un flujo de trabajo no se completan, el estado de ejecución del flujo de trabajo se muestra como `COMPLETED`. Asegúrese de verificar el gráfico de ejecución para detectar los nodos que no se completaron en forma correcta.

- Puede reanudar la ejecución de un flujo de trabajo desde cualquier nodo de trabajo o rastreador que la ejecución de flujo de trabajo original haya intentado ejecutar. No se puede reanudar la ejecución de un flujo de trabajo desde un nodo de desencadenador.
- Al reiniciar un nodo no se restablece su estado. Los datos que se hayan procesado parcialmente no se revertirán.
- Puede reanudar la misma ejecución de flujo de trabajo varias veces. Si una ejecución de flujo de trabajo reanudado sólo se ejecuta en forma parcial, puede solucionar el problema y reanudar la ejecución previamente reanudada.
- Si selecciona dos nodos para reiniciar y dependen uno del otro, el nodo ascendente se ejecuta antes que el nodo descendente. De hecho, seleccionar el nodo descendente es redundante, ya que se ejecutará de acuerdo con los pasos habituales del flujo de trabajo.

Obtención y configuración de propiedades de ejecución de flujo de trabajo en AWS Glue

Utilice propiedades de ejecución de flujo de trabajo para compartir y administrar el estado entre los trabajos en su flujo de trabajo de AWS Glue. Puede definir propiedades de ejecución predeterminadas cuando cree el flujo de trabajo. Por lo tanto, a medida que se ejecutan los trabajos, puede recuperar los valores de propiedad de ejecución y modificarlos opcionalmente para la introducción en trabajos que estén después en el flujo de trabajo. Cuando un trabajo modifica una propiedad de ejecución, el nuevo valor existe solo para la ejecución del flujo de trabajo. Las propiedades de ejecución predeterminadas no se ven afectadas.

Si su trabajo de AWS Glue no forma parte de un flujo de trabajo, estas propiedades no se establecerán.

El siguiente código de Python de ejemplo de un trabajo de extracción, transformación y carga (ETL) muestra cómo obtener las propiedades de ejecución de flujo de trabajo.

```
import sys
import boto3
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from awsglue.context import GlueContext
from pyspark.context import SparkContext

glue_client = boto3.client("glue")
```

```
args = getResolvedOptions(sys.argv, ['JOB_NAME','WORKFLOW_NAME', 'WORKFLOW_RUN_ID'])
workflow_name = args['WORKFLOW_NAME']
workflow_run_id = args['WORKFLOW_RUN_ID']
workflow_params = glue_client.get_workflow_run_properties(Name=workflow_name,
                                          RunId=workflow_run_id)["RunProperties"]

target_database = workflow_params['target_database']
target_s3_location = workflow_params['target_s3_location']
```

El siguiente código continúa con la configuración de la propiedad de ejecución `target_format` en `'csv'`.

```
workflow_params['target_format'] = 'csv'
glue_client.put_workflow_run_properties(Name=workflow_name, RunId=workflow_run_id,
 RunProperties=workflow_params)
```

Para más información, consulte los siguientes temas:

- Acción GetWorkflowRunProperties (Python: get_workflow_run_properties)
- PutWorkflowRunProperties Action (Python: put_workflow_run_properties)

Consulta de flujos de trabajo mediante la AWS Glue API

AWS Glue proporciona una API enriquecida para administrar flujos de trabajo. Puede recuperar una vista estática de un flujo de trabajo o una vista dinámica de un flujo de trabajo en ejecución mediante la AWS Glue API. Para obtener más información, consulte Flujos de trabajo.

Temas

- Consulta de vistas estáticas
- Consulta de vistas dinámicas

Consulta de vistas estáticas

Utilice la operación de la API de `GetWorkflow` para obtener una vista estática que indique el diseño de un flujo de trabajo. Esta operación devuelve un gráfico dirigido que consiste en nodos y perímetros en los que el nodo representa un desencadenador, un trabajo o un rastreador. Los perímetros definen las relaciones entre nodos. Se representan mediante conectores (flechas) en el gráfico en la consola de AWS Glue.

También puede utilizar esta operación con bibliotecas de procesamiento de gráficos conocidas, como NetworkX, igraph, JGraphT y el marco de trabajo Java Universal Network/Graph (JUNG). Puesto que todas estas bibliotecas representan gráficos de forma similar, se precisan transformaciones mínimas.

La vista estática devuelta por esta API es la vista más actualizada según la última definición de desencadenadores asociados a este flujo de trabajo.

Definición de gráfico

Un gráfico de flujo de trabajo G es un par ordenador (N, E), donde N es un conjunto de nodos y E un conjunto de perímetros. Un nodo es un vértice en el gráfico identificado mediante un número único. Un nodo puede ser de tipo desencadenador, trabajo o rastreador. Por ejemplo: `{name:T1, type:Trigger, uniqueId:1}, {name:J1, type:Job, uniqueId:2}`.

Un perímetro es una tupla doble del formulario (`src, dest`), donde `src` y `dest` son los nodos y hay un perímetro dirigido desde `src` a `dest`.

Ejemplo de consulta de una vista estática

Considere un desencadenador condicional T, que desencadena el trabajo J2 una vez que se completa el trabajo J1.

```
J1 ---> T ---> J2
```

Nodos: J1, T, J2

Perímetros: (J1, T), (T, J2)

Consulta de vistas dinámicas

Utilice la operación de la API de `GetWorkflowRun` para obtener una vista dinámica de un flujo de trabajo en ejecución. Esta operación devuelve la misma vista estática del gráfico junto con metadatos relacionados con la ejecución de flujo de datos.

Para la ejecución, los nodos que representan trabajos en la llamada de `GetWorkflowRun` tienen una lista de ejecuciones de trabajo iniciadas como parte de la última ejecución del flujo de trabajo. Puede utilizar esta lista para mostrar el estado de ejecución de cada trabajo en el propio gráfico. Para dependencias descendentes que no se están ejecutando aún, este campo se establece en `null`. La información del gráfico le permitirá conocer el estado actual de cualquier flujo de trabajo en cualquier momento.

La vista dinámica devuelta por esta API se basa en la vista estática que estuviera presente cuando se inició la ejecución de flujo de trabajo.

Ejemplo de nodos de tiempo de ejecución: `{name:T1, type: Trigger, uniqueId:1}`, `{name:J1, type:Job, uniqueId:2, jobDetails:{jobRuns}}`, `{name:C1, type:Crawler, uniqueId:3, crawlerDetails:{crawls}}`

Ejemplo 1: Vista dinámica

En el siguiente ejemplo se muestra un flujo de trabajo de dos desencadenadores simple.

- Nodos: t1, j1, t2, j2
- Perímetros: (t1, j1), (j1, t2), (t2, j2)

La respuesta `GetWorkflow` contiene lo siguiente.

```
{
    Nodes : [
        {
            "type" : Trigger,
            "name" : "t1",
            "uniqueId" : 1
        },
        {
            "type" : Job,
            "name" : "j1",
            "uniqueId" : 2
        },
        {
            "type" : Trigger,
            "name" : "t2",
            "uniqueId" : 3
        },
        {
            "type" : Job,
            "name" : "j2",
            "uniqueId" : 4
        }
    ],
    Edges : [
        {
            "sourceId" : 1,
```

```
            "destinationId" : 2
        },
        {
            "sourceId" : 2,
            "destinationId" : 3
        },
        {
            "sourceId" : 3,
            "destinationId" : 4
        }
}
```

La respuesta `GetWorkflowRun` contiene lo siguiente.

```
{
    Nodes : [
        {
            "type" : Trigger,
            "name" : "t1",
            "uniqueId" : 1,
            "jobDetails" : null,
            "crawlerDetails" : null
        },
        {
            "type" : Job,
            "name" : "j1",
            "uniqueId" : 2,
            "jobDetails" : [
                {
                    "id" : "jr_12334",
                    "jobRunState" : "SUCCEEDED",
                    "errorMessage" : "error string"
                }
            ],
            "crawlerDetails" : null
        },
        {
            "type" : Trigger,
            "name" : "t2",
            "uniqueId" : 3,
            "jobDetails" : null,
            "crawlerDetails" : null
        },
```

```
        {
            "type" : Job,
            "name" : "j2",
            "uniqueId" : 4,
            "jobDetails" : [
                {
                    "id" : "jr_1233sdf4",
                    "jobRunState" : "SUCCEEDED",
                    "errorMessage" : "error string"
                }
            ],
            "crawlerDetails" : null
        }
    ],
    Edges : [
        {
            "sourceId" : 1,
            "destinationId" : 2
        },
        {
            "sourceId" : 2,
            "destinationId" : 3
        },
        {
            "sourceId" : 3,
            "destinationId" : 4
        }
}
```

Ejemplo 2: Varios trabajos con desencadenador condicional

En el siguiente ejemplo se muestra un flujo de trabajo con varios trabajos y un desencadenador condicional (t3).

```
Consider Flow:
T(t1) ---> J(j1) ---> T(t2) ---> J(j2)
             |                    |
             |                    |
             >+------> T(t3) <-----+
                        |
                        |
                      J(j3)
```

```
Graph generated:
Nodes: t1, t2, t3, j1, j2, j3
Edges: (t1, j1), (j1, t2), (t2, j2), (j1, t3), (j2, t3), (t3, j3)
```

Restricciones de esquemas y flujos de trabajo en AWS Glue

A continuación se incluyen las restricciones correspondientes a los proyectos y flujos de trabajo.

Restricciones de esquemas

Tenga en cuenta las siguientes restricciones de proyectos:

- El proyecto debe estar registrado en la misma región de AWS donde reside el bucket de Amazon S3.
- Para compartir proyectos entre cuentas de AWS debe conceder permisos de lectura en el archivo ZIP del proyecto en Amazon S3. Los clientes que tengan permiso de lectura en el archivo ZIP de un proyecto pueden registrar el proyecto en su cuenta de AWS y utilizarlo.
- El conjunto de parámetros del proyecto se almacena como un único objeto JSON. La longitud máxima de este objeto es de 128 KB.
- El tamaño máximo sin comprimir del archivo ZIP del proyecto es de 5 MB. El tamaño máximo de compresión es de 1 MB.
- Limite la cantidad total de trabajos, rastreadores y desencadenadores de un flujo de trabajo a 100 o menos. Si incluye más de 100, es posible que se produzcan errores al intentar reanudar o detener las ejecuciones del flujo de trabajo.

Restricciones del flujo de datos

Tenga en cuenta las siguientes restricciones de flujos de trabajo. Algunos de estos comentarios están dirigidos a un usuario que crea flujos de trabajo en forma manual.

- El tamaño máximo de lote para un desencadenador de eventos de Amazon EventBridge es 100. El tamaño máximo de la ventana es de 900 segundos (15 minutos).
- Un desencadenador se puede asociar solo a un flujo de trabajo.
- Solo se permite un desencadenador de arranque (bajo demanda o de programación).
- Si un desencadenador externo a un flujo de trabajo inicia un trabajo o rastreador dentro del flujo, no se activarán los desencadenadores incluidos en el flujo de trabajo que dependan de la finalización del trabajo o del rastreador (correcta o no).

- Del mismo modo, si un trabajo o rastreador de un flujo de trabajo tiene desencadenadores que dependen de la finalización del trabajo o del rastreador (correcta o no), tanto dentro del flujo de trabajo como fuera de este, si el trabajo o el rastreador se inicia desde dentro de un flujo de trabajo, solo los desencadenadores incluidos en el flujo de trabajo se activan al finalizar el trabajo o el rastreador.

Solución de errores de esquema en AWS Glue

Si detecta errores al utilizar proyectos en AWS Glue, utilice las siguientes soluciones como ayuda para encontrar la fuente de los problemas y corregirlos.

Temas

- Error: falta el módulo PySpark
- Error: falta el archivo de configuración del proyecto
- Error: falta archivo importado
- Error: no autorizado a realizar iamPassRole en el recurso
- Error: programación cron no válida
- Error: ya existe un desencadenador con ese nombre
- Error: el flujo de trabajo con el nombre: foo ya existe.
- Error: no se encontró el módulo en la ruta de LayoutGenerator especificada
- Error: error de validación en el campo Connections (Conexiones)

Error: falta el módulo PySpark

AWS Glue devuelve el error “Unknown error executing layout generator function ModuleNotFoundError: No module named 'pyspark' (Error desconocido al ejecutar la función del generador de diseño ModuleNotFounderError: no hay módulo llamado 'pyspark')”.

Cuando descomprime el archivo del proyecto, podría suceder cualquiera de los dos casos que figuran a continuación:

```
$ unzip compaction.zip
Archive:  compaction.zip
   creating: compaction/
  inflating: compaction/blueprint.cfg
  inflating: compaction/layout.py
```

```
  inflating: compaction/README.md
  inflating: compaction/compaction.py

$ unzip compaction.zip
Archive:  compaction.zip
  inflating: blueprint.cfg
  inflating: compaction.py
  inflating: layout.py
  inflating: README.md
```

En el primer caso, todos los archivos relacionados con el proyecto se colocaron en una carpeta denominada compaction y luego se convirtió en un archivo zip llamado compaction.zip.

En el segundo caso, todos los archivos necesarios para el proyecto no se incluyeron en una carpeta y se agregaron como archivos raíz en el archivo zip compaction.zip.

Se permite crear un archivo en cualquiera de los formatos anteriores. Sin embargo, asegúrese de que `blueprint.cfg` tiene la ruta correcta al nombre de la función en el script que genera el diseño.

Ejemplos

En el caso 1: `blueprint.cfg` debería tener `layoutGenerator` como lo siguiente:

```
layoutGenerator": "compaction.layout.generate_layout"
```

En el caso 2: `blueprint.cfg` debería tener `layoutGenerator` como lo siguiente

```
layoutGenerator": "layout.generate_layout"
```

Si esta ruta no se incluye en forma correcta, podría ver un error como el que se indica. Por ejemplo, si tiene la estructura de carpetas como se menciona en el caso 2 y tiene el `layoutGenerator` indicado como en el caso 1, puede ver el error anterior.

Error: falta el archivo de configuración del proyecto

AWS Glue devuelve el error “Unknown error executing layout generator function FileNotFoundError: [Errno 2] No such file or directory: '/tmp/compaction/blueprint.cfg' (Error desconocido al ejecutar la función del generador de diseño FileNotFoundError: [Errno 2] no existe tal archivo o directorio: '/tmp/compaction/blueprint.cfg')”.

El blueprint.cfg debe colocarse en el nivel raíz del archivo ZIP o dentro de una carpeta que tenga el mismo nombre que el archivo ZIP.

Cuando extraemos el archivo ZIP del proyecto, se espera que blueprint.cfg se encuentre en una de las siguientes rutas. Si no se encuentra en una de las siguientes rutas, puede ver el error anterior.

```
$ unzip compaction.zip
Archive:  compaction.zip
   creating: compaction/
  inflating: compaction/blueprint.cfg

$ unzip compaction.zip
Archive:  compaction.zip
  inflating: blueprint.cfg
```

Error: falta archivo importado

AWS Glue devuelve el error “Unknown error executing layout generator function FileNotFoundError: [Errno 2] No such file or directory:* *'demo-project/foo.py' (Error desconocido al ejecutar la función del generador de diseño FileNotFoundError: [Errno 2] no existe tal archivo o directorio:* *'demo-project/foo.py')”.

Si el script de generación de diseño tiene funcionalidad para leer otros archivos, asegúrese de proporcionar una ruta completa para el archivo que se va a importar. Por ejemplo, se puede hacer referencia al script Conversion.py en Layout.py. Para obtener más información, consulte Esquema de ejemplo.

Error: no autorizado a realizar iamPassRole en el recurso

AWS Glue devuelve el error “User: arn:aws:sts::123456789012:assumed-role/AWSGlueServiceRole/GlueSession is not authorized to perform: iam:PassRole on resource: arn:aws:iam::123456789012:role/AWSGlueServiceRole (Usuario: arn:aws:sts::123456789012:assumed-role/AWSGlueServiceRole/GlueSession no está autorizado para realizar: iam:PassRole en el recurso: arn:aws:iam:: 123456789012:role/AWSGlueServicerole)”

Si los trabajos y rastreadores del flujo de trabajo asumen el mismo rol que el rol transferido para crear el flujo de trabajo a partir del proyecto, el rol del proyecto debe incluir el permiso `iam:PassRole`.

Si los trabajos y rastreadores del flujo de trabajo asumen un rol distinto que el rol transferido para crear las entidades del flujo de trabajo a partir del proyecto, el rol del proyecto debe incluir el permiso `iam:PassRole` en ese otro rol, en lugar de en el rol del proyecto.

Para obtener más información, consulte Permisos para roles de esquema.

Error: programación cron no válida

AWS Glue devuelve el error “The schedule cron(0 0 * * * *) is invalid [El cron de programación (0 0 * * * *) no es válido]”.

Proporcione una expresión cron válida. Para obtener más información, consulte Programaciones basadas en tiempo para trabajos y rastreadores.

Error: ya existe un desencadenador con ese nombre

AWS Glue devuelve el error “Trigger with name 'foo_starting_trigger' already submitted with different configuration (Desencadenador con nombre 'foo_starting_trigger' ya enviado con una configuración diferente)”.

Un proyecto no requiere que defina desencadenadores en el script de diseño para la creación del flujo de trabajo. La creación de desencadenadores es administrada por la biblioteca de proyectos en función de las dependencias definidas entre dos acciones.

La denominación de los desencadenadores es la siguiente:

- Para el desencadenador de inicio en el flujo de trabajo, la denominación es <workflow_name>_starting_trigger.
- Para un nodo (trabajo/rastreador) en el flujo de trabajo que depende de la finalización de uno o varios nodos ascendentes; AWS Glue define un desencadenador con el nombre <workflow_name>_<node_name>_trigger

Este error significa que ya existe un desencadenador con ese nombre. Puede eliminar el desencadenador existente y volver a ejecutar la creación del flujo de trabajo.

> **Note**
>
> Al eliminar un flujo de trabajo no se eliminan los nodos del flujo de trabajo. Es posible que aunque se elimine el flujo de trabajo, los desencadenadores permanezcan. Debido a esto, es posible que no reciba un error de “workflow already exists (el flujo de trabajo ya existe)”, pero puede recibir un error de “trigger already exists (el desencadenador ya existe)” en un caso en el que cree un flujo de trabajo, lo elimine y luego intente volver a crearlo con el mismo nombre y a partir del mismo proyecto.

Error: el flujo de trabajo con el nombre: foo ya existe.

El nombre del flujo de trabajo debe ser único. Inténtelo con un nombre diferente.

Error: no se encontró el módulo en la ruta de LayoutGenerator especificada

AWS Glue devuelve el error “Unknown error executing layout generator function ModuleNotFoundError: No module named 'crawl_s3_locations' (Error desconocido al ejecutar la función del generador de diseño ModuleNotFoundError: no hay módulo llamado 'crawl_s3_locations')”.

```
layoutGenerator": "crawl_s3_locations.layout.generate_layout"
```

Por ejemplo, si tiene la ruta de acceso LayoutGenerator anterior, cuando descomprima el archivo del proyecto, debe tener el siguiente aspecto:

```
$ unzip crawl_s3_locations.zip
Archive:  crawl_s3_locations.zip
   creating: crawl_s3_locations/
  inflating: crawl_s3_locations/blueprint.cfg
  inflating: crawl_s3_locations/layout.py
  inflating: crawl_s3_locations/README.md
```

Cuando descomprime el archivo, si el archivo del proyecto se ve de la siguiente manera, entonces puede obtener el error anterior.

```
$ unzip crawl_s3_locations.zip
Archive:  crawl_s3_locations.zip
  inflating: blueprint.cfg
  inflating: layout.py
  inflating: README.md
```

Puede ver que no hay ninguna carpeta llamada `crawl_s3_locations` y cuando la ruta `layoutGenerator` se refiere al archivo de diseño a través del módulo `crawl_s3_locations`, puede obtener el error anterior.

Error: error de validación en el campo Connections (Conexiones)

AWS Glue devuelve el error “Unknown error executing layout generator function TypeError: Value ['foo'] for key Connections should be of type <class 'dict'>! (Error desconocido al ejecutar la función

del generador de diseño TypeError: el valor ['foo'] para las conexiones clave debe ser de tipo <class 'dict'>!)".

Este es un error de validación. El campo `Connections` en la clase `Job` espera un diccionario y en su lugar se proporciona una lista de valores que causan el error.

```
User input was list of values
Connections= ['string']

Should be a dict like the following
Connections*=*{'Connections': ['string']}
```

Para evitar estos errores de tiempo de ejecución al crear un flujo de trabajo a partir de un proyecto, puede validar las definiciones de flujo de trabajo, trabajo y rastreador como se describe en Realizar pruebas de un esquema.

Consulte la sintaxis en Referencia a clases de proyectos de AWS Glue para definir el trabajo, rastreador y flujo de trabajo de AWS Glue en el script de diseño.

Permisos de personas y roles para esquemas de AWS Glue

Las siguientes son las personas típicas y las políticas de permisos de AWS Identity and Access Management (IAM) sugeridas para personas y roles en relación con esquemas de AWS Glue.

Temas

- Personas para esquemas
- Permisos de personas para esquemas
- Permisos para roles de esquema

Personas para esquemas

Las siguientes son las personas generalmente involucradas en el ciclo de vida de un esquema de AWS Glue.

Persona	Descripción
Desarrollador de AWS Glue	Desarrolla, prueba y publica proyectos.
Administrador de AWS Glue	Registra, mantiene y concede permisos para proyectos.

Persona	Descripción
Analista de datos	Ejecuta proyectos para crear flujos de trabajo.

Para obtener más información, consulte the section called “Información general de los esquemas”.

Permisos de personas para esquemas

Los siguientes son los permisos sugeridos para cada persona del proyecto.

Permisos de desarrollador de AWS Glue para esquemas

El desarrollador de AWS Glue debe tener permisos de escritura en el bucket de Amazon S3 que se utiliza para publicar el esquema. A general, el desarrollador registra el proyecto después de cargarlo. En ese caso, el desarrollador necesita los permisos enumerados en the section called “Permisos de administrador de AWS Glue para esquemas”. Además, si el desarrollador desea probar el proyecto después de su registro, también necesita los permisos enumerados en the section called “Permisos de analista de datos para esquemas”.

Permisos de administrador de AWS Glue para esquemas

La política siguiente concede permisos para registrar, ver y mantener esquemas de AWS Glue.

> **Important**
>
> En la siguiente política , reemplace *<s3-bucket-name>* y *<prefix>* con la ruta de Amazon S3 a los archivos ZIP de proyecto cargados que se registrarán.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:CreateBlueprint",
                "glue:UpdateBlueprint",
                "glue:DeleteBlueprint",
                "glue:GetBlueprint",
                "glue:ListBlueprints",
```

```
                "glue:BatchGetBlueprints"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": "arn:aws:s3:::<s3-bucket-name>/<prefix>/*"
        }
    ]
}
```

Permisos de analista de datos para esquemas

La siguiente política concede permisos para ejecutar proyectos y ver el flujo de trabajo y sus componentes resultantes. También concede `PassRole` al rol que AWS Glue asume para crear el flujo de trabajo y sus componentes.

La política concede permisos sobre cualquier recurso. Si desea configurar el acceso detallado a proyectos individuales, utilice el siguiente formato para los ARN de proyecto:

```
arn:aws:glue:<region>:<account-id>:blueprint/<blueprint-name>
```

> **Important**
>
> En la siguiente política, reemplace *<account-id>* por una cuenta de AWS válida y reemplace *<role-name>* por el nombre del rol utilizado para ejecutar un proyecto. Consulte the section called “Permisos para roles de esquema” para obtener los permisos que requiere este rol.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:ListBlueprints",
                "glue:GetBlueprint",
```

```
                "glue:StartBlueprintRun",
                "glue:GetBlueprintRun",
                "glue:GetBlueprintRuns",
                "glue:GetCrawler",
                "glue:ListTriggers",
                "glue:ListJobs",
                "glue:BatchGetCrawlers",
                "glue:GetTrigger",
                "glue:BatchGetWorkflows",
                "glue:BatchGetTriggers",
                "glue:BatchGetJobs",
                "glue:BatchGetBlueprints",
                "glue:GetWorkflowRun",
                "glue:GetWorkflowRuns",
                "glue:ListCrawlers",
                "glue:ListWorkflows",
                "glue:GetJob",
                "glue:GetWorkflow",
                "glue:StartWorkflowRun"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::<account-id>:role/<role-name>"
        }
    ]
}
```

Permisos para roles de esquema

Los siguientes son los permisos sugeridos para el rol de IAM utilizado para crear un flujo de trabajo a partir de un proyecto. El rol debe tener una relación de confianza con `glue.amazonaws.com`.

> **Important**
>
> En la siguiente política, reemplace *<account-id>* por una cuenta de AWS válida y reemplace *<role-name>* por el nombre del rol.

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:CreateJob",
                "glue:GetCrawler",
                "glue:GetTrigger",
                "glue:DeleteCrawler",
                "glue:CreateTrigger",
                "glue:DeleteTrigger",
                "glue:DeleteJob",
                "glue:CreateWorkflow",
                "glue:DeleteWorkflow",
                "glue:GetJob",
                "glue:GetWorkflow",
                "glue:CreateCrawler"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::<account-id>:role/<role-name>"
        }
    ]
}
```

> **Note**
>
> Si los trabajos y los rastreadores del flujo de trabajo asumen un rol distinto de este rol, esta política debe incluir el permiso `iam:PassRole` en ese otro rol en lugar de en el rol del proyecto.

Desarrollo de esquemas en AWS Glue

Es posible que su organización tenga un conjunto de casos de uso de ETL similares que podrían beneficiarse al poder parametrizar un único flujo de trabajo para manejarlos todos. Para esto, AWS Glue permite definir esquemas, que se pueden utilizar para generar flujos de trabajo. Un proyecto acepta parámetros, de modo que a partir de un único proyecto, un analista de datos puede crear

diferentes flujos de trabajo para manejar casos de uso de ETL similares. Después de crear un proyecto, puede reutilizarlo para diferentes departamentos, equipos y proyectos.

Temas

- Información general de los esquemas en AWS Glue
- Desarrollo de esquemas en AWS Glue
- Registrar un esquema en AWS Glue
- Visualización de esquemas en AWS Glue
- Actualizar un esquema en AWS Glue
- Creación de un flujo de trabajo a partir de un esquema en AWS Glue
- Visualización de las ejecuciones de esquema en AWS Glue

Información general de los esquemas en AWS Glue

> **Note**
>
> En este momento, la característica de esquemas no está disponible en las siguientes regiones de la consola de AWS Glue: Asia Pacífico (Yakarta) y Medio Oriente (Emiratos Árabes Unidos).

Los esquemas de AWS Glue proporcionan una forma de crear y compartir flujos de trabajo de AWS Glue. Cuando hay un proceso de ETL complejo que podría utilizarse para casos de uso similares, en lugar de crear un flujo de trabajo de AWS Glue para cada caso de uso, puede crear un único esquema.

El proyecto especifica los trabajos y rastreadores que se incluirán en un flujo de trabajo, y especifica los parámetros que el usuario del flujo de trabajo proporciona al ejecutar el proyecto para crear un flujo de trabajo. El uso de parámetros permite que un único proyecto genere flujos de trabajo para los distintos casos de uso similares. Para obtener más información acerca de los flujos de trabajo, consulte Información general de los flujos de trabajo en AWS Glue.

A continuación, se muestran ejemplos de casos de uso de proyectos:

- El usuario desea crear una partición de un conjunto de datos existente. Los parámetros de entrada al proyecto son las rutas de origen y destino de Amazon Simple Storage Service (Amazon S3) y una lista de columnas de partición.

- El usuario desea realizar una instantánea de una tabla de Amazon DynamoDB en un almacén de datos SQL como Amazon Redshift. Los parámetros de entrada del esquema son el nombre de la tabla de DynamoDB y una conexión de AWS Glue, que designa un clúster de Amazon Redshift y una base de datos de destino.
- El usuario desea convertir datos CSV en varias rutas de Amazon S3 a Parquet. El usuario desea que el flujo de trabajo de AWS Glue incluya un rastreador y un trabajo independientes para cada ruta. Los parámetros de entrada son la base de datos de destino en el catálogo de datos de AWS Glue y una lista delimitada por comas de rutas de Amazon S3. Tenga en cuenta que, en este caso, el número de rastreadores y trabajos que crea el flujo de trabajo es variable.

Componentes del esquema

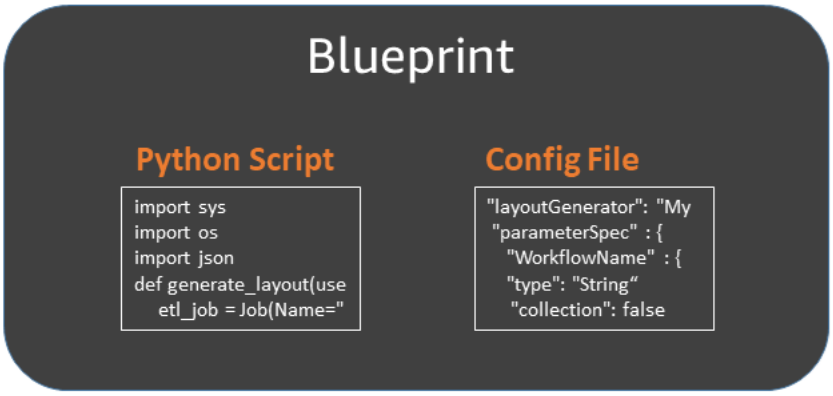
Un proyecto es un archivo ZIP que tiene los siguientes componentes:

- Un script del generador de diseño Python

  Contiene una función que especifica el diseño del flujo de trabajo: los rastreadores y trabajos que se van a crear para el flujo de trabajo, las propiedades del trabajo y del rastreador, y las dependencias entre los trabajos y los rastreadores. La función acepta los parámetros del esquema y devuelve una estructura de flujo de trabajo (objeto JSON) que AWS Glue utiliza para generar el flujo de trabajo. Dado que usted utiliza un script de Python para generar el flujo de trabajo, puede agregar su propia lógica que sea adecuada para sus casos de uso.
- Archivo de configuración

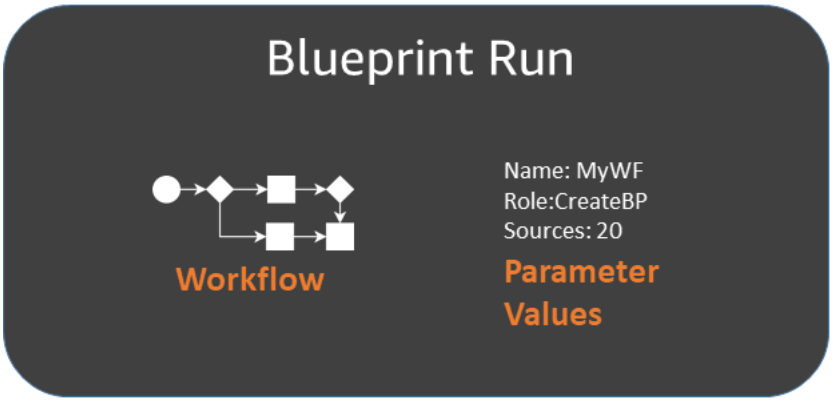
  Especifica el nombre completo de la función de Python que genera el diseño del flujo de trabajo. También especifica los nombres, tipos de datos y otras propiedades de todos los parámetros del proyecto utilizados por el script.
- (Opcional) scripts de ETL y archivos auxiliares

  Como caso de uso avanzado, puede parametrizar la ubicación de los scripts de ETL que utilizan sus trabajos. Puede incluir archivos de script de trabajo en el archivo ZIP y especificar un parámetro del proyecto para una ubicación de Amazon S3 en la que se copiarán los scripts. El script del generador de diseño puede copiar los scripts de ETL en la ubicación designada y especificar esa ubicación como propiedad de ubicación del script de trabajo. También puede incluir bibliotecas u otros archivos auxiliares, siempre que el script los maneje.



Ejecuciones del proyecto

Cuando crea un flujo de trabajo a partir de un esquema, AWS Glue ejecuta el esquema, que inicia un proceso asíncrono para crear el flujo de trabajo y los trabajos, rastreadores y desencadenadores que el flujo de trabajo encapsula. AWS Glue utiliza la ejecución del esquema para orquestar la creación del flujo de trabajo y sus componentes. Para ver el estado del proceso de creación, consulte el estado de ejecución del proyecto. La ejecución del proyecto también almacena los valores que ha proporcionado para los parámetros del proyecto.



Puede visualizar ejecuciones de esquemas con la consola de AWS Glue o AWS Command Line Interface (AWS CLI). Al ver o solucionar problemas de un flujo de trabajo, siempre puede volver a la ejecución del proyecto para ver los valores de parámetro de proyecto que se utilizaron para crear el flujo de trabajo.

Ciclo de vida de un esquema

Los esquemas se desarrollan, prueban y registran con AWS Glue, y se ejecutan para crear flujos de trabajo. En general, hay tres personas involucradas en el ciclo de vida de un proyecto.

Persona	Tareas
Desarrollador de AWS Glue	• Escribe el script de diseño del flujo de trabajo y crea el archivo de configuración. • Realiza pruebas locales del esquema con las bibliotec as proporcionadas por el servicio de AWS Glue. • Crea un archivo ZIP del script, el archivo de configura ción y los archivos auxiliares, y publica el archivo en una ubicación de Amazon S3. • Agrega una política de bucket al bucket de Amazon S3 que concede permisos de lectura en objetos de bucket a la cuenta de AWS del administrador de AWS Glue. • Concede permisos de lectura de IAM en el archivo ZIP de Amazon S3 al administrador de AWS Glue.
Administrador de AWS Glue	• Registra el esquema en AWS Glue. AWS Glue realiza una copia del archivo ZIP en una ubicación reservada de Amazon S3. • Otorga permisos de IAM en el proyecto a los analistas de datos.
Analista de datos	• Ejecuta el proyecto para crear un flujo de trabajo y proporciona valores de parámetros del proyecto. Comprueba el estado de ejecución del proyecto para asegurarse de que el flujo de trabajo y sus component es se generaron en forma correcta. • Ejecuta y soluciona los problemas del flujo de trabajo. Antes de ejecutar el flujo de trabajo, puede verificarlo al acceder al gráfico de diseño del flujo de trabajo en la consola de AWS Glue.

Véase también

- Desarrollo de esquemas en AWS Glue

- Creación de un flujo de trabajo a partir de un esquema en AWS Glue
- Permisos de personas y roles para esquemas de AWS Glue

Desarrollo de esquemas en AWS Glue

Como desarrollador de AWS Glue, puede crear y publicar esquemas que los analistas de datos pueden utilizar para generar flujos de trabajo.

Temas

- Información general sobre el desarrollo de esquemas
- Requisitos previos para desarrollar esquemas
- Escritura del código del esquema
- Proyecto de esquema de ejemplo
- Pruebas a un esquema
- Publicación de un esquema
- Referencia de clases de esquemas de AWS Glue
- Esquemas de ejemplo

Véase también

- Información general de los esquemas en AWS Glue

Información general sobre el desarrollo de esquemas

El primer paso en su proceso de desarrollo es identificar un caso de uso común que se beneficiaría de un proyecto. Un caso de uso típico implica un problema de ETL recurrente que cree que debe resolverse de manera general. Diseñe un proyecto que implemente el caso de uso generalizado y defina los parámetros de entrada del proyecto que, en conjunto, pueden definir un caso de uso específico a partir del caso de uso generalizado.

Un proyecto contiene un archivo de configuración de parámetros del proyecto y un script que define el diseño del flujo de trabajo que se va a generar. El diseño define los trabajos y los rastreadores (o entidades en la terminología del script del proyecto) que se crearán.

No se especifica directamente ningún desencadenador en el script de diseño. En su lugar, se escribe el código para especificar las dependencias entre los trabajos y los rastreadores que crea el script. AWS Glue genera los desencadenadores en función de las especificaciones de dependencia. El resultado del script de diseño es un objeto de flujo de trabajo, que contiene especificaciones para todas las entidades del flujo de trabajo.

Cree el objeto de flujo de trabajo con las siguientes bibliotecas de esquema de AWS Glue:

- `awsglue.blueprint.base_resource`: una biblioteca de recursos básicos utilizados por las bibliotecas.
- `awsglue.blueprint.workflow`: una biblioteca para definir una clase de `Workflow`.
- `awsglue.blueprint.job`: una biblioteca para definir una clase de `Job`.
- `awsglue.blueprint.crawler`: una biblioteca para definir una clase de `Crawler`.

Las únicas otras bibliotecas que se soportan para la generación de diseños son aquellas bibliotecas que están disponibles para el shell de Python.

Antes de publicar el proyecto, puede utilizar los métodos definidos en las bibliotecas del proyecto para probarlo a nivel local.

Cuando esté listo para poner el proyecto a disposición de los analistas de datos, empaquete el script, el archivo de configuración de parámetros y cualquier archivo auxiliar, como scripts y bibliotecas adicionales, en un único activo de implementación. A continuación, cargue el recurso en Amazon S3 y pida a un administrador que lo registre con AWS Glue.

Para obtener más información sobre proyectos de ejemplo, consulte Proyecto de esquema de ejemplo y Esquemas de ejemplo.

Requisitos previos para desarrollar esquemas

Para desarrollar esquemas, debe conocer el uso de AWS Glue y la escritura de scripts para trabajos de ETL de Apache Spark o trabajos de shell de Python. Además, debe completar las siguientes tareas de configuración.

- Descargue cuatro bibliotecas de AWS Python para usar en sus scripts de diseño de proyecto.
- Configure los SDK de AWS.
- Configure AWS CLI.

Descargar las bibliotecas de Python

Descargue las siguientes bibliotecas de GitHub e instálelas en su proyecto:

- https://github.com/awslabs/aws-glue-blueprint-libs/tree/master/awsglue/blueprint/base_resource.py
- https://github.com/awslabs/aws-glue-blueprint-libs/tree/master/awsglue/blueprint/workflow.py
- https://github.com/awslabs/aws-glue-blueprint-libs/tree/master/awsglue/blueprint/crawler.py
- https://github.com/awslabs/aws-glue-blueprint-libs/tree/master/awsglue/blueprint/job.py

Configure el AWS Java SDK

Para el SDK de AWS Java , debe agregar un archivo `jar` que incluya la API para proyectos.

1. Si aún no lo ha hecho, configure el SDK de AWS para Java.
   - Para Java 1.x, siga las instrucciones en Configurar AWS SDK for Java en la Guía para desarrolladores de AWS SDK for Java.
   - Para Java 2.x, siga las instrucciones en Configurar AWS SDK for Java 2.x en la Guía para desarrolladores de AWS SDK for Java 2.x.
2. Descargue el archivo `jar` del cliente que tiene acceso a las API para proyectos.
   - Para Java 1.x: s3: //awsglue-custom-blueprints-preview-artifacts/awsglue-java-sdk-preview/awsglueJavaclient-1.11.x.jar
   - Para Java 2.x: s3: //awsglue-custom-blueprints-preview-artifacts/awsglue-java-sdk-v2-preview/awsjavasdk-glue-2.0.jar
3. Agregue el `jar` del cliente al frente de la ruta de clases de Java para anular el cliente de AWS Glue proporcionado por el SDK de AWS Java.

   ```
   export CLASSPATH=<path-to-preview-client-jar>:$CLASSPATH
   ```
4. (Opcional) pruebe el SDK con la siguiente aplicación Java. La aplicación debe generar una lista vacía.

   Reemplace `accessKey` y `secretKey` con sus credenciales y reemplace `us-east-1` con su región.

   ```
   import com.amazonaws.auth.AWSCredentials;
   import com.amazonaws.auth.AWSCredentialsProvider;
   ```

```
import com.amazonaws.auth.AWSStaticCredentialsProvider;
import com.amazonaws.auth.BasicAWSCredentials;
import com.amazonaws.services.glue.AWSGlue;
import com.amazonaws.services.glue.AWSGlueClientBuilder;
import com.amazonaws.services.glue.model.ListBlueprintsRequest;

public class App{
    public static void main(String[] args) {
        AWSCredentials credentials = new BasicAWSCredentials("accessKey",
 "secretKey");
        AWSCredentialsProvider provider = new
 AWSStaticCredentialsProvider(credentials);
        AWSGlue glue = AWSGlueClientBuilder.standard().withCredentials(provider)
                .withRegion("us-east-1").build();
        ListBlueprintsRequest request = new
 ListBlueprintsRequest().withMaxResults(2);
        System.out.println(glue.listBlueprints(request));
    }
}
```

Configuración de AWS Python SDK

En los siguientes pasos se supone que tiene la versión 2.7 o posterior de Python, o la versión 3.6 o posterior instalada en el equipo.

1. Descargue el siguiente archivo wheel boto3. Si se le solicita abrir o guardar, guarde el archivo. s3: //awsglue-custom-blueprints-preview-artifacts/aws-python-sdk-preview/boto3-1.17.31-py2.py3-none-any.whl
2. Descargue el siguiente archivo wheel de botocore: s3: //awsglue-custom-blueprints-preview-artifacts/aws-python-sdk-preview/botocore-1.20.31-py2.py3-none-any.whl
3. Compruebe su versión de Python.

   ```
   python --version
   ```

4. En función de su versión de Python, ingrese los siguientes comandos (para Linux):

   - Para Python 2.7 o posterior.

     ```
     python3 -m pip install --user virtualenv
     source env/bin/activate
     ```

- Para Python 3.6 o posterior.

```
python3 -m venv python-sdk-test
source python-sdk-test/bin/activate
```

5. Instale el archivo wheel de botocore.

```
python3 -m pip install <download-directory>/botocore-1.20.31-py2.py3-none-any.whl
```

6. Instale el archivo wheel boto3.

```
python3 -m pip install <download-directory>/boto3-1.17.31-py2.py3-none-any.whl
```

7. Configure las credenciales y la región predeterminada en los archivos `~/.aws/credentials` y `~/.aws/config`. Para obtener más información, consulte Configuración de la AWS CLI en la Guía del usuario de AWS Command Line Interface.
8. (Opcional) pruebe su configuración. Los siguientes comandos deben devolver una lista vacía.

   Reemplace la `us-east-1` por su región.

```
$ python
>>> import boto3
>>> glue = boto3.client('glue', 'us-east-1')
>>> glue.list_blueprints()
```

Configuración de la previsualización de AWS CLI

1. Si aún no lo ha hecho, instale o actualice la AWS Command Line Interface (AWS CLI) en su equipo. La manera más sencilla de hacerlo es con `pip`, la utilidad del instalador de Python:

```
pip install awscli --upgrade --user
```

   Puede encontrar instrucciones de instalación completas para AWS CLI aquí: Instalación de AWS Command Line Interface.
2. Descargue el archivo wheel AWS CLI desde: s3: //awsglue-custom-blueprints-preview-artifacts/awscli-preview-build/awscli-1.19.31-py2.py3-none-any.whl
3. Instale el archivo wheel de la AWS CLI.

```
python3 -m pip install awscli-1.19.31-py2.py3-none-any.whl
```

4. Ejecute el comando `aws configure`. Configure sus credenciales de AWS (incluida la clave de acceso y la clave secreta) y la región de AWS. Puede encontrar información acerca de cómo configurar la AWS CLI aquí: Configuración de AWS CLI.
5. Pruebe la AWS CLI. El siguiente comando debe devolver una lista vacía.

   Reemplace la `us-east-1` por su región.

```
aws glue list-blueprints --region us-east-1
```

Escritura del código del esquema

Cada proyecto que cree debe contener como mínimo los siguientes archivos:

- Un script de diseño de Python que define el flujo de trabajo. El script contiene una función que define las entidades (trabajos y rastreadores) en un flujo de trabajo y las dependencias entre ellos.
- Un archivo de configuración, `blueprint.cfg`, que define:
  - La ruta completa de la función de definición de diseño del flujo de trabajo.
  - Los parámetros que acepta el proyecto.

Temas

- Creación del script de diseño del esquema
- Creación del archivo de configuración
- Especificación de parámetros del esquema

Creación del script de diseño del esquema

El script de diseño del proyecto debe incluir una función que genere las entidades en el flujo de trabajo. Puede nombrar esta función como quiera. AWS Glue utiliza el archivo de configuración para determinar el nombre completo de la función.

La función de diseño hace lo siguiente:

- (Opcional) genera instancias de clase `Job` para crear objetos `Job`, y transfiere argumentos como `Command` y `Role`. Estas son propiedades de trabajo que especificaría al crear el trabajo mediante la consola de AWS Glue o la API.
- (Opcional) genera instancias de clase `Crawler` para crear objetos `Crawler`, y transfiere nombre, rol y argumentos de destino.
- Para indicar dependencias entre los objetos (entidades de flujo de trabajo), transfiere los argumentos adicionales `DependsOn` y `WaitForDependencies` a `Job()` y `Crawler()`. Estos argumentos se explican más adelante en esta sección.
- Inicia la clase `Workflow` para crear el objeto de flujo de trabajo que se devuelve a AWS Glue, al transferir un argumento `Name`, un argumento `Entities` y un argumento opcional `OnSchedule`. El argumento `Entities` especifica todos los trabajos y rastreadores que se incluirán en el flujo de trabajo. Para ver cómo construir un objeto `Entities`, consulte el proyecto de ejemplo más adelante en esta sección.
- Devuelve un objeto `Workflow`.

Para conocer las definiciones de las clases `Job`, `Crawler` y `Workflow`, consulte Referencia de clases de esquemas de AWS Glue.

La función de diseño debe aceptar los siguientes argumentos de entrada.

Argumento	Descripción
`user_params`	Diccionario de Python de nombres y valores de parámetros del proyecto. Para obtener más información, consulte Especificación de parámetros del esquema.
`system_params`	Diccionario de Python que contiene dos propiedades: `region` y `accountId` .

Aquí hay un script de generador de diseño de ejemplo en un archivo denominado `Layout.py`:

```
import argparse
import sys
import os
import json
from awsglue.blueprint.workflow import *
```

```
from awsglue.blueprint.job import *
from awsglue.blueprint.crawler import *


def generate_layout(user_params, system_params):

    etl_job = Job(Name="{}_etl_job".format(user_params['WorkflowName']),
                  Command={
                      "Name": "glueetl",
                      "ScriptLocation": user_params['ScriptLocation'],
                      "PythonVersion": "2"
                  },
                  Role=user_params['PassRole'])
    post_process_job = Job(Name="{}_post_process".format(user_params['WorkflowName']),
                           Command={
                               "Name": "pythonshell",
                               "ScriptLocation": user_params['ScriptLocation'],
                               "PythonVersion": "2"
                           },
                           Role=user_params['PassRole'],
                           DependsOn={
                               etl_job: "SUCCEEDED"
                           },
                           WaitForDependencies="AND")
    sample_workflow = Workflow(Name=user_params['WorkflowName'],
                           Entities=Entities(Jobs=[etl_job, post_process_job]))
    return sample_workflow
```

El script de ejemplo importa las bibliotecas del proyecto requeridas e incluye una función `generate_layout` que genera un flujo de trabajo con dos trabajos. Este es un script muy sencillo. Un script más complejo podría emplear lógica y parámetros adicionales para generar un flujo de trabajo con muchos trabajos y rastreadores, o incluso un número variable de trabajos y rastreadores.

Uso del argumento DependsOn (Depende de)

El argumento `DependsOn` es una representación de diccionario de una dependencia que esta entidad tiene en otras entidades dentro del flujo de trabajo. Tiene el formato siguiente.

```
DependsOn = {dependency1 : state, dependency2 : state, ...}
```

Las claves de este diccionario representan la referencia del objeto, no el nombre de la entidad, mientras que los valores son cadenas que corresponden al estado que se debe controlar. AWS Glue

infiere los desencadenantes adecuados. Para conocer los estados válidos, consulte Estructura de condiciones.

Por ejemplo, un trabajo puede depender de la finalización correcta de un rastreador. Si define un objeto de rastreador llamado `crawler2` de la siguiente manera:

```
crawler2 = Crawler(Name="my_crawler", ...)
```

Un objeto que dependa de `crawler2` incluiría un argumento constructor como:

```
DependsOn = {crawler2 : "SUCCEEDED"}
```

Por ejemplo:

```
job1 = Job(Name="Job1", ..., DependsOn = {crawler2 : "SUCCEEDED", ...})
```

Si se omite `DependsOn` para una entidad, esa entidad depende del desencadenador de inicio del flujo de trabajo.

Uso del argumento WaitForDependencies (Esperar a las dependencias)

El argumento `WaitForDependencies` define si una entidad de trabajo o rastreador debe esperar hasta que todas las entidades de las que depende se completen o hasta que alguna se complete.

Los valores permitidos son “AND” o “ANY”.

Uso del argumento OnSchedule (Programado)

El argumento `OnSchedule` para el constructor de clase `Workflow` es una expresión `cron` que establece la definición del desencadenador inicial para un flujo de trabajo.

Si se especifica este argumento, AWS Glue crea un desencadenador de programación con la programación correspondiente. Si no se especifica, el desencadenador de inicio del flujo de trabajo es un desencadenador bajo demanda.

Creación del archivo de configuración

El archivo de configuración del proyecto es un archivo necesario que define el punto de entrada del script para generar el flujo de trabajo, y los parámetros que acepta el proyecto. El archivo debe denominarse `blueprint.cfg`.

A continuación se ofrece un archivo de configuración de ejemplo.

```
{
    "layoutGenerator": "DemoBlueprintProject.Layout.generate_layout",
    "parameterSpec" : {
           "WorkflowName" : {
                "type": "String",
                "collection": false
           },
           "WorkerType" : {
                "type": "String",
                "collection": false,
                "allowedValues": ["G1.X", "G2.X"],
                "defaultValue": "G1.X"
           },
           "Dpu" : {
                "type" : "Integer",
                "allowedValues" : [2, 4, 6],
                "defaultValue" : 2
           },
           "DynamoDBTableName": {
                "type": "String",
                "collection" : false
           },
           "ScriptLocation" : {
                "type": "String",
                "collection": false
     }
    }
}
```

La propiedad `layoutGenerator` especifica el nombre completo de la función en el script que genera el diseño.

La propiedad `parameterSpec` especifica los parámetros que acepta este proyecto. Para obtener más información, consulte Especificación de parámetros del esquema.

> **Important**
>
> El archivo de configuración debe incluir el nombre del flujo de trabajo como parámetro del proyecto, o debe generar un nombre de flujo de trabajo único en el script de diseño.

Especificación de parámetros del esquema

El archivo de configuración contiene especificaciones de parámetros del proyecto en un objeto JSON `parameterSpec`. `parameterSpec` contiene uno o varios objetos de parámetro.

```
"parameterSpec": {
    "<parameter_name>": {
      "type": "<parameter-type>",
      "collection": true|false,
      "description": "<parameter-description>",
      "defaultValue": "<default value for the parameter if value not specified>"
      "allowedValues": "<list of allowed values>"
    },
    "<parameter_name>": {
       ...
    }
  }
```

Las siguientes son las reglas para codificar cada objeto de parámetro:

- El nombre y `type` de parámetro son obligatorios. Todas las demás propiedades son opcionales.
- Si especifica la propiedad `defaultValue`, el parámetro es opcional. De lo contrario, el parámetro es obligatorio y el analista de datos que está creando un flujo de trabajo a partir del proyecto debe proporcionar un valor para él.
- Si establece la propiedad `collection` a `true`, el parámetro puede tomar una recopilación de valores. Las recopilaciones pueden ser de cualquier tipo de datos.
- Si especifica `allowedValues`, la consola de AWS Glue muestra una lista desplegable de valores para que el analista de datos elija al crear un flujo de trabajo a partir del esquema.

Se permiten los siguientes valores para `type`:

Tipos de datos de los parámetros	Notas
`String`	-
`Integer`	-
`Double`	-

Tipos de datos de los parámetros	Notas
`Boolean`	Los posibles valores son `true` y `false`. Genera una casilla de verificación en la página Create a workflow from <blueprint> (Crear un flujo de trabajo a partir de <esquema>) en la consola de AWS Glue.
`S3Uri`	Complete la ruta de Amazon S3. Comience con `s3://`. Genera un campo de texto y el botón Browse (Examinar) en la página Create a workflow from <blueprint> (Crear un flujo de trabajo a partir de <proyecto>).
`S3Bucket`	Solo el nombre del bucket de Amazon S3. Genera un selector de buckets en la página Create a workflow from <blueprint> (Crear un flujo de trabajo a partir de <proyecto>).
`IAMRoleArn`	El nombre de recurso de Amazon (ARN) del rol de AWS Identity and Access Management (IAM). Genera un selector de roles en la página Create a workflow from <blueprint> (Crear un flujo de trabajo a partir de <proyecto>).
`IAMRoleName`	El nombre de un rol de IAM. Genera un selector de roles en la página Create a workflow from <blueprint> (Crear un flujo de trabajo a partir de <proyecto>).

Proyecto de esquema de ejemplo

La conversión de formato de datos es un caso de uso frecuente de extracción, transformación y carga (ETL). En cargas de trabajo analíticas típicas, se prefieren los formatos de archivo basados en columnas como Parquet u ORC sobre los formatos de texto como CSV o JSON. Este proyecto de ejemplo le permite convertir datos de CSV/JSON/etc. a Parquet para archivos en Amazon S3.

Este proyecto toma una lista de rutas S3 definidas por un parámetro del proyecto, convierte los datos al formato Parquet y los escribe en la ubicación S3 especificada por otro parámetro del proyecto. El script de diseño crea un rastreador y un trabajo para cada ruta. El script de diseño también carga el script de ETL en `Conversion.py` a un bucket S3 especificado por otro parámetro del proyecto. Luego, el script de diseño especifica el script cargado como el script de ETL para cada trabajo. El

archivo ZIP del proyecto contiene el script de diseño, el script de ETL y el archivo de configuración del proyecto.

Para obtener más información sobre más proyectos de ejemplo, consulte Esquemas de ejemplo.

El siguiente es el script de diseño en el archivo `Layout.py`.

```
from awsglue.blueprint.workflow import *
from awsglue.blueprint.job import *
from awsglue.blueprint.crawler import *
import boto3

s3_client = boto3.client('s3')

# Ingesting all the S3 paths as Glue table in parquet format
def generate_layout(user_params, system_params):
    #Always give the full path for the file
    with open("ConversionBlueprint/Conversion.py", "rb") as f:
        s3_client.upload_fileobj(f, user_params['ScriptsBucket'], "Conversion.py")
    etlScriptLocation = "s3://{}/Conversion.py".format(user_params['ScriptsBucket'])

    crawlers = []
    jobs = []
    workflowName = user_params['WorkflowName']
    for path in user_params['S3Paths']:
      tablePrefix = "source_"
      crawler = Crawler(Name="{}_crawler".format(workflowName),
                        Role=user_params['PassRole'],
                        DatabaseName=user_params['TargetDatabase'],
                        TablePrefix=tablePrefix,
                        Targets= {"S3Targets": [{"Path": path}]})
      crawlers.append(crawler)
      transform_job = Job(Name="{}_transform_job".format(workflowName),
                          Command={"Name": "glueetl",
                                   "ScriptLocation": etlScriptLocation,
                                   "PythonVersion": "3"},
                          Role=user_params['PassRole'],
                          DefaultArguments={"--database_name":
 user_params['TargetDatabase'],
                                            "--table_prefix": tablePrefix,
                                            "--region_name": system_params['region'],
                                            "--output_path":
 user_params['TargetS3Location']},
                          DependsOn={crawler: "SUCCEEDED"},
```

```
                          WaitForDependencies="AND")
      jobs.append(transform_job)
    conversion_workflow = Workflow(Name=workflowName, Entities=Entities(Jobs=jobs,
 Crawlers=crawlers))
    return conversion_workflow
```

El siguiente es el archivo de configuración del proyecto correspondiente, `blueprint.cfg`.

```
{
    "layoutGenerator": "ConversionBlueprint.Layout.generate_layout",
    "parameterSpec" : {
        "WorkflowName" : {
            "type": "String",
            "collection": false,
            "description": "Name for the workflow."
        },
        "S3Paths" : {
            "type": "S3Uri",
            "collection": true,
            "description": "List of Amazon S3 paths for data ingestion."
        },
        "PassRole" : {
            "type": "IAMRoleName",
            "collection": false,
            "description": "Choose an IAM role to be used in running the job/crawler"
        },
        "TargetDatabase": {
            "type": "String",
            "collection" : false,
            "description": "Choose a database in the Data Catalog."
        },
        "TargetS3Location": {
            "type": "S3Uri",
            "collection" : false,
            "description": "Choose an Amazon S3 output path: ex:s3://<target_path>/."
        },
        "ScriptsBucket": {
            "type": "S3Bucket",
            "collection": false,
            "description": "Provide an S3 bucket name(in the same AWS Region) to store
 the scripts."
        }
    }
```

```
}
```

El siguiente script en el archivo `Conversion.py` es el script de ETL cargado. Tenga en cuenta que el esquema de partición se conserva durante la conversión.

```
import sys
from pyspark.sql.functions import *
from pyspark.context import SparkContext
from awsglue.transforms import *
from awsglue.context import GlueContext
from awsglue.job import Job
from awsglue.utils import getResolvedOptions
import boto3

args = getResolvedOptions(sys.argv, [
    'JOB_NAME',
    'region_name',
    'database_name',
    'table_prefix',
    'output_path'])
databaseName = args['database_name']
tablePrefix = args['table_prefix']
outputPath = args['output_path']

glue = boto3.client('glue', region_name=args['region_name'])

glue_context = GlueContext(SparkContext.getOrCreate())
spark = glue_context.spark_session
job = Job(glue_context)
job.init(args['JOB_NAME'], args)

def get_tables(database_name, table_prefix):
    tables = []
    paginator = glue.get_paginator('get_tables')
    for page in paginator.paginate(DatabaseName=database_name, Expression=table_prefix
+"*"):
        tables.extend(page['TableList'])
    return tables

for table in get_tables(databaseName, tablePrefix):
    tableName = table['Name']
    partitionList = table['PartitionKeys']
    partitionKeys = []
```

```
    for partition in partitionList:
        partitionKeys.append(partition['Name'])

    # Create DynamicFrame from Catalog
    dyf = glue_context.create_dynamic_frame.from_catalog(
        name_space=databaseName,
        table_name=tableName,
        additional_options={
            'useS3ListImplementation': True
        },
        transformation_ctx='dyf'
    )

    # Resolve choice type with make_struct
    dyf = ResolveChoice.apply(
        frame=dyf,
        choice='make_struct',
        transformation_ctx='resolvechoice_' + tableName
    )

    # Drop null fields
    dyf = DropNullFields.apply(
        frame=dyf,
        transformation_ctx="dropnullfields_" + tableName
    )

    # Write DynamicFrame to S3 in glueparquet
    sink = glue_context.getSink(
        connection_type="s3",
        path=outputPath,
        enableUpdateCatalog=True,
        partitionKeys=partitionKeys
    )
    sink.setFormat("glueparquet")

    sink.setCatalogInfo(
        catalogDatabase=databaseName,
        catalogTableName=tableName[len(tablePrefix):]
    )
    sink.writeFrame(dyf)

job.commit()
```

> **Note**
>
> Sólo se pueden suministrar dos rutas de Amazon S3 como entrada al proyecto de ejemplo. Esto se debe a que los desencadenadores de AWS Glue se limitan a invocar solo dos acciones de rastreador.

Pruebas a un esquema

Mientras desarrolla el código, debe realizar pruebas locales para verificar que el diseño del flujo de trabajo es correcto.

Las pruebas locales no generan trabajos, rastreadores o desencadenadores de AWS Glue. En su lugar, ejecute el script de diseño a nivel local y utilice los métodos `to_json()` y `validate()` para imprimir objetos y buscar errores. Estos métodos están disponibles en las tres clases definidas en las bibliotecas.

Hay dos formas de manejar los argumentos `user_params` y `system_params` que AWS Glue transfiere a la función de diseño. Su código de banco de pruebas puede crear un diccionario de valores de parámetros del proyecto de ejemplo y transferirlo a la función de diseño como el argumento `user_params`. O bien, puede eliminar las referencias a `user_params` y reemplazarlas con cadenas codificadas.

Si su código hace uso de las propiedades `region` y `accountId` en el argumento `system_params`, puede transferir su propio diccionario para `system_params`.

Para realizar pruebas a un proyecto

1. Inicie un intérprete de Python en un directorio con las bibliotecas, o cargue los archivos del proyecto y las bibliotecas suministradas en su entorno de desarrollo integrado (IDE) preferido.
2. Asegúrese de que el código importe las bibliotecas suministradas.
3. Agregue código a su función de diseño para realizar llamadas a `validate()` o `to_json()` en cualquier entidad o en el objeto de `Workflow`. Por ejemplo, si su código crea un objeto de `Crawler` denominado `mycrawler`, puede llamar a `validate()` de la siguiente manera.

   ```
   mycrawler.validate()
   ```

   Puede imprimir `mycrawler` de la siguiente manera:

```
print(mycrawler.to_json())
```

Si llama a `to_json` en un objeto, no hay necesidad de llamar también a `validate()`, ya que `to_json()` llama a `validate()`.

Es muy útil llamar a estos métodos en el objeto de flujo de trabajo. Suponiendo que el script nombra el objeto de flujo de trabajo `my_workflow`, valide e imprima el objeto de flujo de trabajo de la siguiente manera.

```
print(my_workflow.to_json())
```

Para obtener más información sobre `to_json()` y `validate()`, consulte Métodos de clase.

También puede importar `pprint` e imprimir el objeto de flujo de trabajo, tal y como se muestra en el ejemplo que se encuentra a continuación en esta sección.

4. Ejecute el código, corrija los errores y, por último, elimine cualquier llamada a `validate()` o `to_json()`.

Example

El siguiente ejemplo muestra cómo construir un diccionario de parámetros del proyecto de ejemplo y transferirlo como el argumento `user_params` a la función de diseño `generate_compaction_workflow`. También muestra cómo imprimir el objeto de flujo de trabajo generado.

```
from pprint import pprint
from awsglue.blueprint.workflow import *
from awsglue.blueprint.job import *
from awsglue.blueprint.crawler import *

USER_PARAMS = {"WorkflowName": "compaction_workflow",
               "ScriptLocation": "s3://awsexamplebucket1/scripts/threaded-
compaction.py",
               "PassRole": "arn:aws:iam::111122223333:role/GlueRole-ETL",
               "DatabaseName": "cloudtrial",
               "TableName": "ct_cloudtrail",
               "CoalesceFactor": 4,
               "MaxThreadWorkers": 200}
```

```
def generate_compaction_workflow(user_params: dict, system_params: dict) -> Workflow:
    compaction_job = Job(Name=f"{user_params['WorkflowName']}_etl_job",
                         Command={"Name": "glueetl",
                                  "ScriptLocation": user_params['ScriptLocation'],
                                  "PythonVersion": "3"},
                         Role="arn:aws:iam::111122223333:role/
AWSGlueServiceRoleDefault",
                         DefaultArguments={"DatabaseName": user_params['DatabaseName'],
                                           "TableName": user_params['TableName'],
                                           "CoalesceFactor":
 user_params['CoalesceFactor'],
                                           "max_thread_workers":
 user_params['MaxThreadWorkers']})

    catalog_target = {"CatalogTargets": [{"DatabaseName": user_params['DatabaseName'],
 "Tables": [user_params['TableName']]}]}

    compacted_files_crawler = Crawler(Name=f"{user_params['WorkflowName']}_post_crawl",
                                      Targets = catalog_target,
                                      Role=user_params['PassRole'],
                                      DependsOn={compaction_job: "SUCCEEDED"},
                                      WaitForDependencies="AND",
                                      SchemaChangePolicy={"DeleteBehavior": "LOG"})

    compaction_workflow = Workflow(Name=user_params['WorkflowName'],
                                   Entities=Entities(Jobs=[compaction_job],

 Crawlers=[compacted_files_crawler]))
    return compaction_workflow

generated = generate_compaction_workflow(user_params=USER_PARAMS, system_params={})
gen_dict = generated.to_json()

pprint(gen_dict)
```

Publicación de un esquema

Después de crear un proyecto, debe cargarlo en Amazon S3. Debe disponer de permisos de escritura en el bucket de Amazon S3 que utilice para publicar el proyecto. También debe asegurarse de que el administrador de AWS Glue, que registrará el esquema, dispone de acceso de lectura al bucket de Amazon S3. Para las políticas de permiso de AWS Identity and Access Management (IAM)

sugeridas para personas y roles para esquemas de AWS Glue, consulte Permisos de personas y roles para esquemas de AWS Glue.

Para publicar un proyecto

1. Cree los scripts, recursos y archivo de configuración del proyecto necesarios.
2. Agregue todos los archivos a un archivo ZIP y cargue el archivo ZIP a Amazon S3. Utilice un bucket de S3 que se encuentre en la misma región que la región en la que los usuarios se registrarán y ejecutarán el proyecto.

   Puede crear un archivo ZIP desde la línea de comandos con el siguiente comando.

   ```
   zip -r folder.zip folder
   ```

3. Agregue una política de bucket que otorgue permisos de lectura a la cuenta deseada de AWS. La siguiente es una política de ejemplo.

   ```
   {
       "Version": "2012-10-17",
       "Statement": [
           {
               "Effect": "Allow",
               "Principal": {
                   "AWS": "arn:aws:iam::111122223333:root"
               },
               "Action": "s3:GetObject",
               "Resource": "arn:aws:s3:::my-blueprints/*"
           }
       ]
   }
   ```

4. Conceda el permiso `s3:GetObject` de IAM en el bucket de Amazon S3 al administrador de AWS Glue o a quien registrará los esquemas. Para obtener una política de ejemplo para conceder a los administradores, consulte Permisos de administrador de AWS Glue para esquemas.

Después de haber completado las pruebas locales del esquema, es posible que también desee realizar pruebas a un esquema en AWS Glue. Para realizar pruebas a un esquema en AWS Glue, este debe estar registrado. Puede restringir quién verá el proyecto registrado mediante la autorización de IAM o mediante cuentas de prueba independientes.

Véase también:

- Registrar un esquema en AWS Glue

Referencia de clases de esquemas de AWS Glue

Las bibliotecas para los esquemas de AWS Glue definen tres clases que se utilizan en el script de diseño del flujo de trabajo: `Job`, `Crawler` y `Workflow`.

Temas

- Clase de trabajo
- Clase de rastreador
- Clase de flujo de trabajo
- Métodos de clase

Clase de trabajo

La clase de `Job` representa un trabajo de ETL de AWS Glue.

Argumentos obligatorios del constructor

A continuación, se indican los argumentos obligatorios del constructor para la clase de `Job`.

Nombre del argumento	Tipo	Descripción
`Name`	`str`	Nombre a asignar al trabajo. AWS Glue agrega un sufijo generado en forma aleatoria al nombre para distinguir el trabajo de los creados por otras ejecuciones del esquema.
`Role`	`str`	El nombre de recurso de Amazon (ARN) del rol que el trabajo debe asumir mientras se ejecuta.
`Command`	`dict`	Comando de trabajo, tal como se especifica en JobCommand estructura en la documentación de la API.

Argumentos opcionales del constructor

A continuación, se indican los argumentos opcionales del constructor para la clase de `Job`.

Nombre del argumento	Tipo	Descripción
`DependsOn`	`dict`	Lista de entidades de flujo de trabajo de las que depende el trabajo. Para obtener más información, consulte Uso del argumento DependsOn (Depende de).
`WaitForDependencies`	`str`	Indica si el trabajo debe esperar hasta que todas las entidades de las que depende se completen antes de ejecutarse o hasta que alguna se complete. Para obtener más información, consulte Uso del argumento WaitForDependencies (Esperar a las dependencias). Omitir si el trabajo depende de una sola entidad.
(Propiedades del trabajo)	-	Cualquiera de las propiedades del trabajo enumeradas en Estructura de trabajo en la documentación de la API de AWS Glue (excepto `CreatedOn` y `LastModifiedOn`).

Clase de rastreador

La clase de `Crawler` representa un rastreador de AWS Glue.

Argumentos obligatorios del constructor

A continuación, se indican los argumentos obligatorios del constructor para la clase de `Crawler`.

Nombre del argumento	Tipo	Descripción
`Name`	`str`	Nombre a asignar al rastreador. AWS Glue agrega un sufijo generado en forma aleatoria al nombre para distinguir el rastreador de los creados por otras ejecuciones del esquema.

Nombre del argumento	Tipo	Descripción
`Role`	`str`	ARN del rol que el rastreador debe asumir durante la ejecución.
`Targets`	`dict`	Recopilación de destinos que se rastreará n. Los argumentos del constructor de clase de `Targets` se definen en CrawlerTargets estructura en la documentación de la API. Todos los argumentos del constructor de `Targets` son opcionales, pero debe transferir al menos uno.

Argumentos opcionales del constructor

A continuación, se indican los argumentos opcionales del constructor para la clase de `Crawler`.

Nombre del argumento	Tipo	Descripción
`DependsOn`	`dict`	Lista de entidades del flujo de trabajo de las que depende el rastreador. Para obtener más información, consulte Uso del argumento DependsOn (Depende de).
`WaitForDependencies`	`str`	Indica si el rastreador debe esperar hasta que todas las entidades de las que depende se completen antes de ejecutarse o hasta que alguna se complete. Para obtener más informaci ón, consulte Uso del argumento WaitForDependencies (Esperar a las dependencias). Omitir si el rastreador depende de una sola entidad.
(Propiedades del rastreador)	-	Cualquiera de las propiedades del rastreador enumeradas en Estructura de rastreador en la documentación de la API de AWS Glue, con las siguientes excepciones: • `State`

Nombre del argumento	Tipo	Descripción
		• `CrawlElapsedTime` • `CreationTime` • `LastUpdated` • `LastCrawl` • `Version`

Clase de flujo de trabajo

La clase de `Workflow` representa un flujo de trabajo de AWS Glue. El script de diseño del flujo de trabajo devuelve un objeto `Workflow`. AWS Gluecrea un flujo de trabajo basado en este objeto.

Argumentos obligatorios del constructor

A continuación, se indican los argumentos obligatorios del constructor para la clase de `Workflow`.

Nombre del argumento	Tipo	Descripción
`Name`	`str`	El nombre que se asignará al flujo de trabajo.
`Entities`	`Entities`	Conjunto de entidades (trabajos y rastreadores) que se incluirán en el flujo de trabajo. El constructor de clase de `Entities` acepta un argumento de `Jobs`, que es una lista de objetos de `Job`, y un argumento de `Crawlers`, que es una lista de objetos de `Crawler`.

Argumentos opcionales del constructor

A continuación, se indican los argumentos opcionales del constructor para la clase de `Workflow`.

Nombre del argumento	Tipo	Descripción
`Description`	`str`	Consulte Estructura de flujo de trabajo.

Nombre del argumento	Tipo	Descripción
`DefaultRunProperties`	`dict`	Consulte Estructura de flujo de trabajo.
`OnSchedule`	`str`	Una expresión `cron`.

Métodos de clase

Las tres clases incluyen los métodos siguientes.

validate() [validar()]

Valida las propiedades del objeto y, si se encuentran errores, genera un mensaje y sale. No genera resultados si no hay errores. Para la clase de `Workflow`, se llama a sí mismo en cada entidad en el flujo de trabajo.

to_json() [a_json()]

Serializa el objeto a JSON. También llama a `validate()`. Para la clase de `Workflow`, el objeto JSON incluye listas de trabajos y rastreadores, y una lista de desencadenadores generados por las especificaciones de dependencia del trabajo y del rastreador.

Esquemas de ejemplo

Hay una serie de esquemas de ejemplo disponibles en el repositorio de Github de esquemas de AWS Glue. Estas muestras son solo de referencia y no están destinadas a su uso en producción.

Los títulos de los proyectos de ejemplo son:

- Compaction (Compactación): este proyecto crea un trabajo que compacta los archivos de entrada en segmentos más grandes en función del tamaño de archivo deseado.
- Conversion (Conversión): este proyecto convierte archivos de entrada en varios formatos de archivo estándar en formato Apache Parquet, que está optimizado para cargas de trabajo analíticas.
- Crawling Amazon S3 locations (Rastreo de ubicaciones de Amazon S3): este proyecto rastrea varias ubicaciones de Amazon S3 para agregar tablas de metadatos al Data Catalog.
- Custom connection to Data Catalog (Conexión personalizada a Data Catalog): este esquema accede a los almacenes de datos mediante conectores personalizados de AWS Glue, lee los

registros y completa las definiciones de tabla en el catálogo de datos de AWS Glue en función del esquema de registros.

- Encoding (Codificación): este proyecto convierte sus archivos no UTF en archivos codificados UTF.
- Partitioning (Particionamiento): este proyecto crea un trabajo de partición que coloca los archivos de salida en particiones en función de claves de partición específicas.
- Importing Amazon S3 data into a DynamoDB table (Importación de datos de Amazon S3 en una tabla de DynamoDB): este proyecto importa datos de Amazon S3 en una tabla de DynamoDB.
- Standard table to governed (Tabla estándar que regirá): este esquema importa una tabla de Data Catalog de AWS Glue a una tabla de Lake Formation.

Registrar un esquema en AWS Glue

Después de que el desarrollador de AWS Glue ha codificado el esquema y ha cargado un archivo ZIP a Amazon Simple Storage Service (Amazon S3), un administrador de AWS Glue debe registrar el esquema. Registrar el proyecto hace que esté disponible para su uso.

Cuando registra un esquema, AWS Glue copia el archivo de esquemas en una ubicación reservada de Amazon S3. Podrá eliminar el archivo de la ubicación de carga.

Para registrar un proyecto, necesita permisos de lectura en la ubicación de Amazon S3 que contiene el archivo cargado. También necesita el permiso de AWS Identity and Access Management (IAM), `glue:CreateBlueprint`. Para obtener los permisos sugeridos para un administrador de AWS Glue que debe registrar, ver y mantener los esquemas, consulte Permisos de administrador de AWS Glue para esquemas.

Puede registrar un esquema con la consola de AWS Glue, la API de AWS Glue o AWS Command Line Interface (AWS CLI).

Para registrar un proyecto (consola)

1. Asegúrese de que dispone de permisos de lectura (`s3:GetObject`) en el archivo ZIP del proyecto en Amazon S3.
2. Abra la consola de AWS Glue en https://console.aws.amazon.com/glue/.

   Inicie sesión como usuario con permisos para registrar un proyecto. Cambie a la misma región de AWS que el bucket de Amazon S3 que contiene el archivo ZIP del proyecto.

3. En el panel de navegación, elija Blueprints (Esquemas). Luego, en la página Blueprints (Esquemas), elija Add blueprint (Agregar esquema).
4. Escriba un nombre de proyecto y una descripción opcional.
5. Para ZIP archive location (S3) [Ubicación del archivo ZIP (S3)], ingrese la ruta de Amazon S3 del archivo ZIP del proyecto que se subió. Incluya el nombre del archivo en la ruta y comience la ruta con `s3://`.
6. (Opcional) agregue una o más etiquetas.
7. Elija Add blueprint (Agregar proyecto).

   La página Blueprints (Esquemas) vuelve a aparecer y muestra que el estado del proyecto es `CREATING`. Elija el botón de actualización hasta que el estado cambie a `ACTIVE` o `FAILED`.
8. Si el estado es `FAILED`, seleccione el proyecto y, en el menú Actions (Acciones), elija View (Visualizar).

   En la página de detalles se muestra el motivo del error. Si el mensaje de error es “Unable to access object at location... (No se puede acceder al objeto en la ubicación...)” o “Access denied on object at location... (Acceso denegado al objeto en la ubicación...)”, revise los siguientes requisitos:

   - El usuario con el que ha iniciado sesión debe tener permiso de lectura en el archivo ZIP del proyecto en Amazon S3.
   - El bucket de Amazon S3 que contiene el archivo ZIP debe tener una política de bucket que otorgue permiso de lectura sobre el objeto a su ID de cuenta de AWS. Para obtener más información, consulte Desarrollo de esquemas en AWS Glue.
   - El bucket de Amazon S3 que está utilizando debe estar en la misma región que la región en la que ha iniciado sesión en la consola.
9. Asegúrese de que los analistas de datos tengan permisos sobre el proyecto.

   La política de IAM sugerida para analistas de datos se muestra en Permisos de analista de datos para esquemas. Esta política concede `glue:GetBlueprint` sobre cualquier recurso. Si su política está más detallada al nivel de recursos, otorgue permisos a los analistas de datos sobre este recurso recién creado.

Para registrar un proyecto (AWS CLI)

1. Escriba el siguiente comando.

```
aws glue create-blueprint --name <blueprint-name> [--description <description>] --
blueprint-location s3://<s3-path>/<archive-filename>
```

2. Ingrese el siguiente comando para comprobar el estado del proyecto. Repita el comando hasta que el estado cambie a `ACTIVE` o `FAILED`.

```
aws glue get-blueprint --name <blueprint-name>
```

Si el estado es `FAILED` y el mensaje de error es “Unable to access object at location... (No se puede acceder al objeto en la ubicación...)” o “Access denied on object at location... (Acceso denegado al objeto en la ubicación...)”, revise los siguientes requisitos:

- El usuario con el que ha iniciado sesión debe tener permiso de lectura en el archivo ZIP del proyecto en Amazon S3.
- El bucket de Amazon S3 que contiene el archivo ZIP debe tener una política de bucket que otorgue permiso de lectura sobre el objeto a su ID de cuenta de AWS. Para obtener más información, consulte Publicación de un esquema.
- El bucket de Amazon S3 que está utilizando debe estar en la misma región que la región en la que ha iniciado sesión en la consola.

Véase también:

- Información general de los esquemas en AWS Glue

Visualización de esquemas en AWS Glue

Visualice un proyecto para revisar la descripción, el estado y las especificaciones de parámetros del proyecto, y para descargar el archivo ZIP del proyecto.

Puede visualizar un esquema con la consola de AWS Glue, la API de AWS Glue o AWS Command Line Interface (AWS CLI).

Para visualizar un proyecto (consola)

1. Abra la consola de AWS Glue en https://console.aws.amazon.com/glue/.

2. En el panel de navegación, elija Blueprints (Esquemas).
3. En la página Blueprints (Esquemas), seleccione un esquema. Luego, en el menú Actions (Acciones), seleccione View (Visualización).

Para visualizar un proyecto (AWS CLI)

- Ingrese el siguiente comando para ver sólo el nombre, la descripción y el estado del proyecto. Reemplace *<blueprint-name>* con el nombre del proyecto que se va a visualizar.

```
aws glue get-blueprint --name <blueprint-name>
```

El resultado del comando es similar al siguiente.

```
{
    "Blueprint": {
        "Name": "myDemoBP",
        "CreatedOn": 1587414516.92,
        "LastModifiedOn": 1587428838.671,
        "BlueprintLocation": "s3://awsexamplebucket1/demo/
DemoBlueprintProject.zip",
        "Status": "ACTIVE"
    }
}
```

Ingrese el siguiente comando para ver también las especificaciones del parámetro.

```
aws glue get-blueprint --name <blueprint-name>  --include-parameter-spec
```

El resultado del comando es similar al siguiente.

```
{
    "Blueprint": {
        "Name": "myDemoBP",
        "CreatedOn": 1587414516.92,
        "LastModifiedOn": 1587428838.671,
        "ParameterSpec": "{\"WorkflowName\":{\"type\":\"String\",\"collection
\":false,\"description\":null,\"defaultValue\":null,\"allowedValues\":null},
\"PassRole\":{\"type\":\"String\",\"collection\":false,\"description\":null,
\"defaultValue\":null,\"allowedValues\":null},\"DynamoDBTableName\":{\"type
```

```
\":\"String\",\"collection\":false,\"description\":null,\"defaultValue\":null,
\"allowedValues\":null},\"ScriptLocation\":{\"type\":\"String\",\"collection
\":false,\"description\":null,\"defaultValue\":null,\"allowedValues\":null}}",
        "BlueprintLocation": "s3://awsexamplebucket1/demo/
DemoBlueprintProject.zip",
        "Status": "ACTIVE"
    }
}
```

Agregue el argumento `--include-blueprint` a fin de incluir una URL en el resultado que puede pegar en el navegador para descargar el archivo ZIP del esquema que AWS Glue almacenó.

Véase también:

- Información general de los esquemas en AWS Glue

Actualizar un esquema en AWS Glue

Puede actualizar un proyecto si tiene un script de diseño revisado, un conjunto revisado de parámetros del proyecto o archivos de soporte revisados. Al actualizar un proyecto, se crea una versión nueva.

La actualización de un proyecto no afecta los flujos de trabajo existentes creados a partir del proyecto.

Puede actualizar un esquema con la consola de AWS Glue, la API de AWS Glue o AWS Command Line Interface (AWS CLI).

En el siguiente procedimiento se supone que el desarrollador de AWS Glue ha creado y cargado un archivo ZIP del esquema actualizado en Amazon S3.

Para actualizar un proyecto (consola)

1. Asegúrese de que dispone de permisos de lectura (`s3:GetObject`) en el archivo ZIP del proyecto en Amazon S3.
2. Abra la consola de AWS Glue en https://console.aws.amazon.com/glue/.

Inicie sesión como usuario con permisos para actualizar un proyecto. Cambie a la misma región de AWS que el bucket de Amazon S3 que contiene el archivo ZIP del proyecto.

3. En el panel de navegación, elija Blueprints (Esquemas).
4. En la página Blueprints (Esquemas), seleccione un proyecto y, en el menú Actions (Acciones), elija Edit (Editar).
5. En la página Edit a blueprint (Editar un proyecto), actualice la Description (Descripción) o ZIP archive location (S3) [Ubicación del archivo ZIP (S3)] del proyecto. Asegúrese de incluir el nombre del archivo en la ruta.
6. Elija Save (Guardar).

   La página Blueprints (Esquemas) vuelve a aparecer y muestra que el estado del proyecto es `UPDATING`. Elija el botón de actualización hasta que el estado cambie a `ACTIVE` o `FAILED`.
7. Si el estado es `FAILED`, seleccione el proyecto y, en el menú Actions (Acciones), elija View (Visualizar).

   En la página de detalles se muestra el motivo del error. Si el mensaje de error es “Unable to access object at location... (No se puede acceder al objeto en la ubicación...)” o “Access denied on object at location... (Acceso denegado al objeto en la ubicación...)”, revise los siguientes requisitos:

   - El usuario con el que ha iniciado sesión debe tener permiso de lectura en el archivo ZIP del proyecto en Amazon S3.
   - El bucket de Amazon S3 que contiene el archivo ZIP debe tener una política de bucket que otorgue permiso de lectura sobre el objeto a su ID de cuenta de AWS. Para obtener más información, consulte Publicación de un esquema.
   - El bucket de Amazon S3 que está utilizando debe estar en la misma región que la región en la que ha iniciado sesión en la consola.

> **Note**
>
> Si se produce un error en la actualización, la siguiente ejecución del proyecto utiliza la versión más reciente del proyecto que se registró o actualizó correctamente.

Para actualizar un proyecto (AWS CLI)

1. Ingrese el siguiente comando.

```
aws glue update-blueprint --name <blueprint-name> [--description <description>] --
blueprint-location s3://<s3-path>/<archive-filename>
```

2. Ingrese el siguiente comando para comprobar el estado del proyecto. Repita el comando hasta que el estado cambie a `ACTIVE` o `FAILED`.

```
aws glue get-blueprint --name <blueprint-name>
```

   Si el estado es `FAILED` y el mensaje de error es “Unable to access object at location... (No se puede acceder al objeto en la ubicación...)” o “Access denied on object at location... (Acceso denegado al objeto en la ubicación...)”, revise los siguientes requisitos:

   - El usuario con el que ha iniciado sesión debe tener permiso de lectura en el archivo ZIP del proyecto en Amazon S3.
   - El bucket de Amazon S3 que contiene el archivo ZIP debe tener una política de bucket que otorgue permiso de lectura sobre el objeto a su ID de cuenta de AWS. Para obtener más información, consulte Publicación de un esquema.
   - El bucket de Amazon S3 que está utilizando debe estar en la misma región que la región en la que ha iniciado sesión en la consola.

> Véase también
>
> - Información general de los esquemas en AWS Glue

Creación de un flujo de trabajo a partir de un esquema en AWS Glue

Puede crear un flujo de trabajo de AWS Glue en forma manual al agregar un componente a la vez, o puede crear un flujo de trabajo desde un esquema de AWS Glue. AWS Glue incluye esquemas para casos de uso habituales. Los desarrolladores de AWS Glue pueden crear esquemas adicionales.

> **Important**
>
> Limite la cantidad total de trabajos, rastreadores y desencadenadores de un flujo de trabajo a 100 o menos. Si incluye más de 100, es posible que se produzcan errores al intentar reanudar o detener las ejecuciones del flujo de trabajo.

Cuando utiliza un esquema, puede generar rápidamente un flujo de trabajo para un caso de uso específico basado en el caso de uso generalizado definido por el esquema. Defina el caso de uso específico al proporcionar valores para los parámetros del esquema. Por ejemplo, un esquema que particiona un conjunto de datos podría tener como parámetros las rutas de origen y destino de Amazon S3.

AWS Glue crea un flujo de trabajo a partir de un esquema al ejecutar el esquema. La ejecución del esquema guarda los valores de parámetro proporcionados y se utiliza para realizar un seguimiento del progreso y el resultado de la creación del flujo de trabajo y sus componentes. Al solucionar problemas de un flujo de trabajo, puede visualizar la ejecución del esquema a fin de determinar los valores de los parámetros del esquema que se utilizaron para crear el flujo de trabajo.

Para crear y visualizar flujos de trabajo, necesita determinados permisos de IAM. Si desea ver una política de IAM sugerida, consulte Permisos de analista de datos para esquemas.

Puede crear un flujo de trabajo a partir de un esquema mediante la consola de AWS Glue, la API de AWS Glue o la AWS Command Line Interface (AWS CLI).

Para crear un flujo de trabajo a partir de un esquema (consola)

1. Abra la consola de AWS Glue en https://console.aws.amazon.com/glue/.

   Inicie sesión como usuario con permisos para crear un flujo de trabajo.
2. En el panel de navegación, elija Blueprints (Esquemas).
3. Seleccione un esquema y, en el menú Actions (Acciones), elija Create workflow (Crear flujo de trabajo).
4. En la página Create a workflow from <blueprint-name> (Crear un flujo de trabajo a partir de <nombre del esquema>), ingrese la siguiente información:

Parámetros del esquema

Estos varían en función del diseño del esquema. Si tiene preguntas sobre los parámetros, consulte al desarrollador. Los esquemas generalmente incluyen un parámetro para el nombre del flujo de trabajo.

Rol de IAM

El rol que asume AWS Glue para crear el flujo de trabajo y sus componentes. El rol debe tener permisos para crear y eliminar flujos de trabajo, trabajos, rastreadores y desencadenadores. Para obtener una política sugerida para el rol, consulte Permisos para roles de esquema.

5. Elija Enviar.

   Aparece la página Blueprint details (Detalles del esquema), que muestra una lista de ejecuciones del esquema en la parte inferior.

6. En la lista de ejecuciones del esquema, verifique el estado de creación del flujo de trabajo en la última ejecución del esquema.

   El estado inicial es.`RUNNING` Elija el botón de actualización hasta que el estado pase a `SUCCEEDED` o `FAILED`.

7. Realice alguna de las siguientes acciones:

   - Si el estado de finalización es `SUCCEEDED`, puede ir a la página Workflows (Flujos de trabajo), seleccionar el flujo de trabajo recién creado y ejecutarlo. Antes de ejecutar el flujo de trabajo, puede revisar el gráfico de diseño.
   - Si el estado de finalización es `FAILED`, seleccione la ejecución del esquema y, en el menú Actions (Acciones), elija View (Visualizar) para ver el mensaje de error.

Para obtener más información sobre flujos de trabajo y esquemas, consulte los siguientes temas.

- Información general de los flujos de trabajo en AWS Glue
- Actualizar un esquema en AWS Glue
- Creación y desarrollo manual de un flujo de trabajo en AWS Glue

Visualización de las ejecuciones de esquema en AWS Glue

Visualice una ejecución de proyecto para ver la siguiente información:

- Nombre del flujo de trabajo que se creó.
- Valores de parámetros del esquema que se utilizaron para crear el flujo de trabajo.
- El estado de la operación de creación del flujo de trabajo.

Puede visualizar la ejecución de un esquema con la consola de AWS Glue, la API de AWS Glue o AWS Command Line Interface (AWS CLI).

Para visualizar una ejecución del proyecto (consola)

1. Abra la consola de AWS Glue en https://console.aws.amazon.com/glue/.
2. En el panel de navegación, elija Blueprints (Esquemas).
3. En la página Blueprints (Esquemas), seleccione un esquema. Luego, en el menú Actions (Acciones), seleccione View (Visualización).
4. En la parte inferior de la página Blueprint Details (Detalles del proyecto), seleccione una ejecución de proyecto y, en el menú Actions (Acciones), elija View (Visualizaciones).

Para visualizar la ejecución de un proyecto (AWS CLI)

- Ingrese el siguiente comando. Reemplace *<blueprint-name>* con el nombre del proyecto. Reemplace *<blueprint-run-id>* con el ID de ejecución del proyecto.

```
aws glue get-blueprint-run --blueprint-name <blueprint-name> --run-id <blueprint-run-id>
```

Véase también:

- Información general de los esquemas en AWS Glue

AWS CloudFormation para AWS Glue

AWS CloudFormation es un servicio que puede crear muchos recursos de AWS. AWS Glue proporciona operaciones de API para crear objetos en AWS Glue Data Catalog. Sin embargo, podría ser más cómodo definir y crear objetos de AWS Glue y otros objetos de recursos de AWS relacionados en un archivo de plantilla de AWS CloudFormation. A continuación, puede automatizar el proceso de creación de objetos.

AWS CloudFormation proporciona una sintaxis simplificada, JSON (JavaScript Object Notation) o YAML (YAML Ain't Markup Language), para expresar la creación de recursos de AWS. Puede utilizar las plantillas de AWS CloudFormation para definir objetos del Catálogo de datos, como bases de datos, tablas, particiones, rastreadores, clasificadores y conexiones. También puede definir objetos ETL, por ejemplo, trabajos, disparadores y puntos de enlace de desarrollo. Puede crear una plantilla que describa todos los recursos de AWS que desea y AWS CloudFormation se encargará del aprovisionamiento y la configuración de dichos recursos.

Para obtener más información, consulte ¿Qué es AWS CloudFormation? y Trabajar con plantillas AWS CloudFormation en la Guía del usuario de AWS CloudFormation.

Si tiene previsto utilizar plantillas de AWS CloudFormation que sean compatibles con AWS Glue, como administrador, debe conceder acceso a AWS CloudFormation y a los servicios y acciones de AWS de los que depende. Para otorgar permisos para crear recursos de AWS CloudFormation, adjunte la siguiente política a los usuarios que trabajan con AWS CloudFormation:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudformation:*"
      ],
      "Resource": "*"
    }
  ]
}
```

La siguiente tabla contiene las acciones que puede realizar una plantilla de AWS CloudFormation en su nombre. Incluye enlaces a información sobre los tipos de recursos de AWS y sus tipos de propiedad que puede agregar a una plantilla de AWS CloudFormation.

Recurso de AWS Glue	Plantilla de AWS CloudFormation	Ejemplos de AWS Glue
Clasificador	AWS::Glue::Classifier	Clasificador de grok, clasificador de JSON, clasificador de XML
Connection	AWS::Glue::Connection	Conexión de MySQL
Rastreador	AWS::Glue::Crawler	Rastreador de Amazon S3, rastreador de MySQL
Base de datos	AWS::Glue::Database	Base de datos vacía, base de datos con tablas
Punto de enlace de desarrollo	AWS::Glue::DevEndpoint	Punto de enlace de desarrollo
Trabajo	AWS::Glue::Job	Trabajo de Amazon S3, trabajo de JDBC
Transformación de machine learning	AWS::Glue::MLTransform	Transformación de machine learning
Conjunto de reglas de calidad de datos	AWS::Glue::DataQualityRuleset	Conjunto de reglas de calidad de datos, conjunto de reglas de calidad de datos con el programador de EventBridge
Partición	AWS::Glue::Partition	Particiones de una tabla
Tabla	AWS::Glue::Table	Tabla en una base de datos
Desencadenador	AWS::Glue::Trigger	Disparador bajo demanda, disparador programado, disparador condicional

Para comenzar, utilice las siguientes plantillas de muestra y personalícelas con sus propios metadatos. A continuación, utilice la consola de AWS CloudFormation para crear una pila de AWS CloudFormation con el fin de agregar objetos a AWS Glue y cualquier servicio asociado. Muchos campos de un objeto de AWS Glue son opcionales. Estas plantillas ilustran los campos que son obligatorios o necesarios para un objeto de AWS Glue operativo y funcional.

Una plantilla de AWS CloudFormation puede tener formato JSON o YAML. En estos ejemplos, se utiliza YAML para facilitar la legibilidad. Los ejemplos contienen comentarios (#) para describir los valores que están definidos en las plantillas.

Las plantillas de AWS CloudFormation pueden incluir una sección `Parameters`. Esta sección se puede modificar en el texto de muestra o cuando se envía el archivo YAML a la consola de AWS CloudFormation para crear una pila. La sección `Resources` de la plantilla contiene la definición de AWS Glue y los objetos relacionados. Las definiciones de sintaxis de plantilla de AWS CloudFormation puede contener propiedades que incluyan sintaxis de propiedad más detallada. Es posible que no todas las propiedades sean necesarias para crear un objeto de AWS Glue. Estas muestras incluyen valores de ejemplo de propiedades comunes para crear un objeto de AWS Glue.

Plantilla de AWS CloudFormation de muestra para una base de datos de AWS Glue

Una base de datos de AWS Glue en el Catálogo de datos contiene tablas de metadatos. La base de datos se compone de muy pocas propiedades y se puede crear en el Catálogo de datos con una plantilla de AWS CloudFormation. La siguiente plantilla de muestra se proporciona para ayudarle a comenzar y para ilustrar el uso de las pilas de AWS CloudFormation con AWS Glue. El único recurso creado por la plantilla de muestra es una base de datos denominada `cfn-mysampledatabase`. Puede cambiarlo si edita el texto de la muestra o cambia el valor en la consola de AWS CloudFormation al enviar el código YAML.

A continuación, se muestran valores de ejemplo de propiedades comunes para crear una base de datos de AWS Glue. Para obtener más información sobre la plantilla de base de datos de AWS CloudFormation para AWS Glue, consulte AWS::Glue::Database.

```
---
AWSTemplateFormatVersion: '2010-09-09'
# Sample CloudFormation template in YAML to demonstrate creating a database named
 mysampledatabase
```

```
# The metadata created in the Data Catalog points to the flights public S3 bucket
#
# Parameters section contains names that are substituted in the Resources section
# These parameters are the names the resources created in the Data Catalog
Parameters:
  CFNDatabaseName:
    Type: String
    Default: cfn-mysampledatabse

# Resources section defines metadata for the Data Catalog
Resources:
# Create an AWS Glue database
  CFNDatabaseFlights:
    Type: AWS::Glue::Database
    Properties:
      # The database is created in the Data Catalog for your account
      CatalogId: !Ref AWS::AccountId
      DatabaseInput:
        # The name of the database is defined in the Parameters section above
        Name: !Ref CFNDatabaseName
        Description: Database to hold tables for flights data
        LocationUri: s3://crawler-public-us-east-1/flight/2016/csv/
        #Parameters: Leave AWS database parameters blank
```

Plantilla de AWS CloudFormation de muestra para una base de datos, tabla y partición de AWS Glue

Una tabla de AWS Glue contiene los metadatos que definen la estructura y la ubicación de los datos que desea procesar con los scripts de ETL. En una tabla, puede definir particiones para paralelizar el procesamiento de los datos. Una partición es un fragmento de datos que define con una clave. Por ejemplo, si se usa el mes como una clave, todos los datos de enero se encuentran en la misma partición. En AWS Glue, las bases de datos pueden contener tablas y las tablas pueden contener particiones.

En la siguiente muestra se presenta cómo rellenar una base de datos, una tabla y particiones con una plantilla de AWS CloudFormation. El formato de los datos base es `csv` y están delimitados por una coma (,). Como debe existir una base de datos para que pueda contener una tabla, y una tabla debe existir para que se puedan crear particiones, la plantilla utiliza la instrucción `DependsOn` para definir la dependencia de estos objetos cuando se crean.

Los valores de esta muestra definen una tabla que contiene los datos de vuelos de un bucket de Amazon S3 disponible públicamente. Con fines ilustrativos, solo hay definidas algunas columnas de los datos y una clave de partición. También se definen cuatro particiones en el Catálogo de datos. En los campos `StorageDescriptor` también se muestran algunos campos para describir el almacenamiento de los datos base.

```
---
AWSTemplateFormatVersion: '2010-09-09'
# Sample CloudFormation template in YAML to demonstrate creating a database, a table,
 and partitions
# The metadata created in the Data Catalog points to the flights public S3 bucket
#
# Parameters substituted in the Resources section
# These parameters are names of the resources created in the Data Catalog
Parameters:
  CFNDatabaseName:
    Type: String
    Default: cfn-database-flights-1
  CFNTableName1:
    Type: String
    Default: cfn-manual-table-flights-1
# Resources to create metadata in the Data Catalog
Resources:
###
# Create an AWS Glue database
  CFNDatabaseFlights:
    Type: AWS::Glue::Database
    Properties:
      CatalogId: !Ref AWS::AccountId
      DatabaseInput:
        Name: !Ref CFNDatabaseName
        Description: Database to hold tables for flights data
###
# Create an AWS Glue table
  CFNTableFlights:
    # Creating the table waits for the database to be created
    DependsOn: CFNDatabaseFlights
    Type: AWS::Glue::Table
    Properties:
      CatalogId: !Ref AWS::AccountId
      DatabaseName: !Ref CFNDatabaseName
      TableInput:
```

```
        Name: !Ref CFNTableName1
        Description: Define the first few columns of the flights table
        TableType: EXTERNAL_TABLE
        Parameters: {
    "classification": "csv"
  }
#       ViewExpandedText: String
        PartitionKeys:
        # Data is partitioned by month
        - Name: mon
          Type: bigint
        StorageDescriptor:
          OutputFormat: org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat
          Columns:
          - Name: year
            Type: bigint
          - Name: quarter
            Type: bigint
          - Name: month
            Type: bigint
          - Name: day_of_month
            Type: bigint
          InputFormat: org.apache.hadoop.mapred.TextInputFormat
          Location: s3://crawler-public-us-east-1/flight/2016/csv/
          SerdeInfo:
            Parameters:
              field.delim: ","
            SerializationLibrary: org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe
# Partition 1
# Create an AWS Glue partition
  CFNPartitionMon1:
    DependsOn: CFNTableFlights
    Type: AWS::Glue::Partition
    Properties:
      CatalogId: !Ref AWS::AccountId
      DatabaseName: !Ref CFNDatabaseName
      TableName: !Ref CFNTableName1
      PartitionInput:
        Values:
        - 1
        StorageDescriptor:
          OutputFormat: org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat
          Columns:
          - Name: mon
```

```
            Type: bigint
          InputFormat: org.apache.hadoop.mapred.TextInputFormat
          Location: s3://crawler-public-us-east-1/flight/2016/csv/mon=1/
          SerdeInfo:
            Parameters:
              field.delim: ","
            SerializationLibrary: org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe
# Partition 2
# Create an AWS Glue partition
  CFNPartitionMon2:
    DependsOn: CFNTableFlights
    Type: AWS::Glue::Partition
    Properties:
      CatalogId: !Ref AWS::AccountId
      DatabaseName: !Ref CFNDatabaseName
      TableName: !Ref CFNTableName1
      PartitionInput:
        Values:
        - 2
        StorageDescriptor:
          OutputFormat: org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat
          Columns:
          - Name: mon
            Type: bigint
          InputFormat: org.apache.hadoop.mapred.TextInputFormat
          Location: s3://crawler-public-us-east-1/flight/2016/csv/mon=2/
          SerdeInfo:
            Parameters:
              field.delim: ","
            SerializationLibrary: org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe
# Partition 3
# Create an AWS Glue partition
  CFNPartitionMon3:
    DependsOn: CFNTableFlights
    Type: AWS::Glue::Partition
    Properties:
      CatalogId: !Ref AWS::AccountId
      DatabaseName: !Ref CFNDatabaseName
      TableName: !Ref CFNTableName1
      PartitionInput:
        Values:
        - 3
        StorageDescriptor:
          OutputFormat: org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat
```

```
          Columns:
          - Name: mon
            Type: bigint
          InputFormat: org.apache.hadoop.mapred.TextInputFormat
          Location: s3://crawler-public-us-east-1/flight/2016/csv/mon=3/
          SerdeInfo:
            Parameters:
              field.delim: ","
            SerializationLibrary: org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe
# Partition 4
# Create an AWS Glue partition
  CFNPartitionMon4:
    DependsOn: CFNTableFlights
    Type: AWS::Glue::Partition
    Properties:
      CatalogId: !Ref AWS::AccountId
      DatabaseName: !Ref CFNDatabaseName
      TableName: !Ref CFNTableName1
      PartitionInput:
        Values:
        - 4
        StorageDescriptor:
          OutputFormat: org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat
          Columns:
          - Name: mon
            Type: bigint
          InputFormat: org.apache.hadoop.mapred.TextInputFormat
          Location: s3://crawler-public-us-east-1/flight/2016/csv/mon=4/
          SerdeInfo:
            Parameters:
              field.delim: ","
            SerializationLibrary: org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe
```

Plantilla de AWS CloudFormation de muestra para un clasificador de grok de AWS Glue

Un clasificador de AWS Glue determina el esquema de sus datos. Un tipo de clasificador personalizado utiliza un patrón de grok para adaptarse a sus datos. Si el patrón coincide, el clasificador personalizado se utiliza para crear el esquema de la tabla y se establece `classification` en el valor configurado en la definición de clasificador.

Esta muestra genera un clasificador que crea un esquema con una columna denominada `message` y establece la clasificación en `greedy`.

```
---
AWSTemplateFormatVersion: '2010-09-09'
# Sample CFN YAML to demonstrate creating a classifier
#
# Parameters section contains names that are substituted in the Resources section
# These parameters are the names the resources created in the Data Catalog
Parameters:

# The name of the classifier to be created
  CFNClassifierName:
    Type: String
    Default: cfn-classifier-grok-one-column-1

#
#
# Resources section defines metadata for the Data Catalog
Resources:
# Create classifier that uses grok pattern to put all data in one column and classifies
 it as "greedy".
  CFNClassifierFlights:
    Type: AWS::Glue::Classifier
    Properties:
      GrokClassifier:
        #Grok classifier that puts all data in one column
        Name: !Ref CFNClassifierName
        Classification: greedy

        GrokPattern: "%{GREEDYDATA:message}"
        #CustomPatterns: none
```

Plantilla de AWS CloudFormation de muestra para un clasificador de JSON de AWS Glue

Un clasificador de AWS Glue determina el esquema de sus datos. Un tipo de clasificador personalizado utiliza una cadena `JsonPath` que define los datos JSON que el clasificador debe clasificar. AWS Glue es compatible con un subconjunto de los operadores de `JsonPath`, tal y como se describe en Escritura de clasificadores personalizados JSON.

Si el patrón coincide, el clasificador personalizado se utiliza para crear el esquema de la tabla.

En esta muestra se crea un clasificador que genera un esquema con cada registro de la matriz `Records3` en un objeto.

```
---
AWSTemplateFormatVersion: '2010-09-09'
# Sample CFN YAML to demonstrate creating a JSON classifier
#
# Parameters section contains names that are substituted in the Resources section
# These parameters are the names the resources created in the Data Catalog
Parameters:

# The name of the classifier to be created
  CFNClassifierName:
    Type: String
    Default: cfn-classifier-json-one-column-1

#
#
# Resources section defines metadata for the Data Catalog
Resources:
# Create classifier that uses a JSON pattern.
  CFNClassifierFlights:
    Type: AWS::Glue::Classifier
    Properties:
      JSONClassifier:
        #JSON classifier
        Name: !Ref CFNClassifierName
        JsonPath: $.Records3[*]
```

Plantilla de AWS CloudFormation de muestra para un clasificador de XML de AWS Glue

Un clasificador de AWS Glue determina el esquema de sus datos. Un tipo de clasificador personalizado especifica una etiqueta XML para designar el elemento que contiene cada registro en un documento XML que se está analizando. Si el patrón coincide, el clasificador personalizado se utiliza para crear el esquema de la tabla y se establece `classification` en el valor configurado en la definición de clasificador.

En esta muestra se crea un clasificador que genera un esquema con cada registro de la etiqueta `Record` y establece la clasificación en `XML`.

```
---
AWSTemplateFormatVersion: '2010-09-09'
# Sample CFN YAML to demonstrate creating an XML classifier
#
# Parameters section contains names that are substituted in the Resources section
# These parameters are the names the resources created in the Data Catalog
Parameters:

# The name of the classifier to be created
  CFNClassifierName:
    Type: String
    Default: cfn-classifier-xml-one-column-1

#
#
# Resources section defines metadata for the Data Catalog
Resources:
# Create classifier that uses the XML pattern and classifies it as "XML".
  CFNClassifierFlights:
    Type: AWS::Glue::Classifier
    Properties:
      XMLClassifier:
        #XML classifier
        Name: !Ref CFNClassifierName
        Classification: XML
        RowTag: <Records>
```

Plantilla de AWS CloudFormation de muestra para un rastreador de AWS Glue para Amazon S3

Un rastreador de AWS Glue crea tablas de metadatos en el Catálogo de datos que se corresponden con sus datos. A continuación, puede utilizar estas definiciones de tabla como orígenes y destinos en sus trabajos de ETL.

Esta muestra crea un rastreador, el rol de IAM necesario y una base de datos de AWS Glue en el Catálogo de datos. Cuando se ejecuta este rastreador, asume el rol de IAM y crea una tabla en la base de datos correspondiente a los datos de vuelos públicos. La tabla se crea con

el prefijo "`cfn_sample_1_`". El rol de IAM que crea esta plantilla otorga permisos globales; es recomendable que cree un rol personalizado. No se definen clasificadores personalizados mediante este clasificador. De forma predeterminada, se usan los clasificadores integrados de AWS Glue.

Al enviar esta muestra a la consola de AWS CloudFormation, debe confirmar que desea crear el rol de IAM.

```
---
AWSTemplateFormatVersion: '2010-09-09'
# Sample CFN YAML to demonstrate creating a crawler
#
# Parameters section contains names that are substituted in the Resources section
# These parameters are the names the resources created in the Data Catalog
Parameters:

# The name of the crawler to be created
  CFNCrawlerName:
    Type: String
    Default: cfn-crawler-flights-1
  CFNDatabaseName:
    Type: String
    Default: cfn-database-flights-1
  CFNTablePrefixName:
    Type: String
    Default: cfn_sample_1_
#
#
# Resources section defines metadata for the Data Catalog
Resources:
#Create IAM Role assumed by the crawler. For demonstration, this role is given all
 permissions.
  CFNRoleFlights:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
          -
            Effect: "Allow"
            Principal:
              Service:
                - "glue.amazonaws.com"
```

```
            Action:
              - "sts:AssumeRole"
      Path: "/"
      Policies:
        -
          PolicyName: "root"
          PolicyDocument:
            Version: "2012-10-17"
            Statement:
              -
                Effect: "Allow"
                Action: "*"
                Resource: "*"
 # Create a database to contain tables created by the crawler
  CFNDatabaseFlights:
    Type: AWS::Glue::Database
    Properties:
      CatalogId: !Ref AWS::AccountId
      DatabaseInput:
        Name: !Ref CFNDatabaseName
        Description: "AWS Glue container to hold metadata tables for the flights
 crawler"
 #Create a crawler to crawl the flights data on a public S3 bucket
  CFNCrawlerFlights:
    Type: AWS::Glue::Crawler
    Properties:
      Name: !Ref CFNCrawlerName
      Role: !GetAtt CFNRoleFlights.Arn
      #Classifiers: none, use the default classifier
      Description: AWS Glue crawler to crawl flights data
      #Schedule: none, use default run-on-demand
      DatabaseName: !Ref CFNDatabaseName
      Targets:
        S3Targets:
          # Public S3 bucket with the flights data
          - Path: "s3://crawler-public-us-east-1/flight/2016/csv"
      TablePrefix: !Ref CFNTablePrefixName
      SchemaChangePolicy:
        UpdateBehavior: "UPDATE_IN_DATABASE"
        DeleteBehavior: "LOG"
      Configuration: "{\"Version\":1.0,\"CrawlerOutput\":{\"Partitions\":
{\"AddOrUpdateBehavior\":\"InheritFromTable\"},\"Tables\":{\"AddOrUpdateBehavior\":
\"MergeNewColumns\"}}}"
```

Plantilla de AWS CloudFormation de muestra para una conexión de AWS Glue

Una conexión de AWS Glue en el Catálogo de datos contiene la información de JDBC y de red que es necesaria para conectarse a una base de datos JDBC. Esta información se utiliza al conectarse a una base de datos JDBC para rastrear o ejecutar trabajos de ETL.

Esta muestra crea una conexión a una base de datos de Amazon RDS MySQL denominada `devdb`. Cuando se utiliza esta conexión, también se suministran un rol de IAM, credenciales de base de datos y valores de conexión de red. Consulte los detalles de los campos necesarios en la plantilla.

```
---
AWSTemplateFormatVersion: '2010-09-09'
# Sample CFN YAML to demonstrate creating a connection
#
# Parameters section contains names that are substituted in the Resources section
# These parameters are the names the resources created in the Data Catalog
Parameters:

# The name of the connection to be created
  CFNConnectionName:
    Type: String
    Default: cfn-connection-mysql-flights-1
  CFNJDBCString:
    Type: String
    Default: "jdbc:mysql://xxx-mysql.yyyyyyyyyyyyyy.us-east-1.rds.amazonaws.com:3306/
devdb"
  CFNJDBCUser:
    Type: String
    Default: "master"
  CFNJDBCPassword:
    Type: String
    Default: "12345678"
    NoEcho: true
#
#
# Resources section defines metadata for the Data Catalog
Resources:
  CFNConnectionMySQL:
    Type: AWS::Glue::Connection
    Properties:
```

```
      CatalogId: !Ref AWS::AccountId
      ConnectionInput:
        Description: "Connect to MySQL database."
        ConnectionType: "JDBC"
        #MatchCriteria: none
        PhysicalConnectionRequirements:
          AvailabilityZone: "us-east-1d"
          SecurityGroupIdList:
           - "sg-7d52b812"
          SubnetId: "subnet-84f326ee"
        ConnectionProperties: {
          "JDBC_CONNECTION_URL": !Ref CFNJDBCString,
          "USERNAME": !Ref CFNJDBCUser,
          "PASSWORD": !Ref CFNJDBCPassword
        }
        Name: !Ref CFNConnectionName
```

Plantilla de AWS CloudFormation de muestra para un rastreador de AWS Glue para JDBC

Un rastreador de AWS Glue crea tablas de metadatos en el Catálogo de datos que se corresponden con sus datos. A continuación, puede utilizar estas definiciones de tabla como orígenes y destinos en sus trabajos de ETL.

Esta muestra crea un rastreador, el rol de IAM necesario y una base de datos de AWS Glue en el Catálogo de datos. Cuando se ejecuta este rastreador, asume el rol de IAM y crea una tabla en la base de datos correspondiente a los datos de vuelos públicos que se han almacenado en una base de datos MySQL. La tabla se crea con el prefijo "`cfn_jdbc_1_`". El rol de IAM que crea esta plantilla otorga permisos globales; es recomendable que cree un rol personalizado. No se puede definir ningún clasificador personalizado para los datos de JDBC. De forma predeterminada, se usan los clasificadores integrados de AWS Glue.

Al enviar esta muestra a la consola de AWS CloudFormation, debe confirmar que desea crear el rol de IAM.

```
---
AWSTemplateFormatVersion: '2010-09-09'
# Sample CFN YAML to demonstrate creating a crawler
#
```

```
# Parameters section contains names that are substituted in the Resources section
# These parameters are the names the resources created in the Data Catalog
Parameters:

# The name of the crawler to be created
  CFNCrawlerName:
    Type: String
    Default: cfn-crawler-jdbc-flights-1
# The name of the database to be created to contain tables
  CFNDatabaseName:
    Type: String
    Default: cfn-database-jdbc-flights-1
# The prefix for all tables crawled and created
  CFNTablePrefixName:
    Type: String
    Default: cfn_jdbc_1_
# The name of the existing connection to the MySQL database
  CFNConnectionName:
    Type: String
    Default: cfn-connection-mysql-flights-1
# The name of the JDBC path (database/schema/table) with wildcard (%) to crawl
  CFNJDBCPath:
    Type: String
    Default: saldev/%
#
#
# Resources section defines metadata for the Data Catalog
Resources:
#Create IAM Role assumed by the crawler. For demonstration, this role is given all
 permissions.
  CFNRoleFlights:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
          -
            Effect: "Allow"
            Principal:
              Service:
                - "glue.amazonaws.com"
            Action:
              - "sts:AssumeRole"
      Path: "/"
```

```
      Policies:
        -
          PolicyName: "root"
          PolicyDocument:
            Version: "2012-10-17"
            Statement:
              -
                Effect: "Allow"
                Action: "*"
                Resource: "*"
 # Create a database to contain tables created by the crawler
  CFNDatabaseFlights:
    Type: AWS::Glue::Database
    Properties:
      CatalogId: !Ref AWS::AccountId
      DatabaseInput:
        Name: !Ref CFNDatabaseName
        Description: "AWS Glue container to hold metadata tables for the flights
 crawler"
 #Create a crawler to crawl the flights data in MySQL database
  CFNCrawlerFlights:
    Type: AWS::Glue::Crawler
    Properties:
      Name: !Ref CFNCrawlerName
      Role: !GetAtt CFNRoleFlights.Arn
      #Classifiers: none, use the default classifier
      Description: AWS Glue crawler to crawl flights data
      #Schedule: none, use default run-on-demand
      DatabaseName: !Ref CFNDatabaseName
      Targets:
        JdbcTargets:
          # JDBC MySQL database with the flights data
          - ConnectionName: !Ref CFNConnectionName
            Path: !Ref CFNJDBCPath
          #Exclusions: none
      TablePrefix: !Ref CFNTablePrefixName
      SchemaChangePolicy:
        UpdateBehavior: "UPDATE_IN_DATABASE"
        DeleteBehavior: "LOG"
    Configuration: "{\"Version\":1.0,\"CrawlerOutput\":{\"Partitions\":
{\"AddOrUpdateBehavior\":\"InheritFromTable\"},\"Tables\":{\"AddOrUpdateBehavior\":
\"MergeNewColumns\"}}}"
```

Plantilla de AWS CloudFormation de muestra para un trabajo de AWS Glue para Amazon S3 a Amazon S3

Un trabajo de AWS Glue en el Catálogo de datos contiene los valores de parámetros que son necesarios para ejecutar un script en AWS Glue.

Esta muestra crea un trabajo que lee datos de vuelos de un bucket de Amazon S3 en formato `csv` y los escribe en un archivo Parquet de Amazon S3. El script que ejecuta este flujo de trabajo ya debe existir. Puede generar un script de ETL para su entorno con la consola de AWS Glue. Cuando este trabajo se ejecuta, también se debe proporcionar un rol de IAM con los permisos correspondientes.

Los valores de parámetro comunes se muestran en la plantilla. Por ejemplo, el valor predeterminado de `AllocatedCapacity` (DPU) es 5.

```
---
AWSTemplateFormatVersion: '2010-09-09'
# Sample CFN YAML to demonstrate creating a job using the public flights S3 table in a
 public bucket
#
# Parameters section contains names that are substituted in the Resources section
# These parameters are the names the resources created in the Data Catalog
Parameters:

# The name of the job to be created
  CFNJobName:
    Type: String
    Default: cfn-job-S3-to-S3-2
# The name of the IAM role that the job assumes. It must have access to data, script,
 temporary directory
  CFNIAMRoleName:
    Type: String
    Default: AWSGlueServiceRoleGA
# The S3 path where the script for this job is located
  CFNScriptLocation:
    Type: String
    Default: s3://aws-glue-scripts-123456789012-us-east-1/myid/sal-job-test2
#
#
# Resources section defines metadata for the Data Catalog
Resources:
```

```
# Create job to run script which accesses flightscsv table and write to S3 file as
 parquet.
# The script already exists and is called by this job
  CFNJobFlights:
    Type: AWS::Glue::Job
    Properties:
      Role: !Ref CFNIAMRoleName
      #DefaultArguments: JSON object
      # If script written in Scala, then set DefaultArguments={'--job-language';
 'scala', '--class': 'your scala class'}
      #Connections:  No connection needed for S3 to S3 job
      #  ConnectionsList
      #MaxRetries: Double
      Description: Job created with CloudFormation
      #LogUri: String
      Command:
        Name: glueetl
        ScriptLocation: !Ref CFNScriptLocation
             # for access to directories use proper IAM role with permission to buckets
 and folders that begin with "aws-glue-"
             # script uses temp directory from job definition if required (temp
 directory not used S3 to S3)
             # script defines target for output as s3://aws-glue-target/sal
      AllocatedCapacity: 5
      ExecutionProperty:
        MaxConcurrentRuns: 1
      Name: !Ref CFNJobName
```

Plantilla de AWS CloudFormation de muestra para un trabajo de AWS Glue para JDBC a Amazon S3

Un trabajo de AWS Glue en el Catálogo de datos contiene los valores de parámetros que son necesarios para ejecutar un script en AWS Glue.

Esta muestra crea un trabajo que lee los datos de vuelos de una base de datos MySQL JDBC según lo definido por la conexión denominada `cfn-connection-mysql-flights-1` y los escribe en un archivo Parquet de Amazon S3. El script que ejecuta este flujo de trabajo ya debe existir. Puede generar un script de ETL para su entorno con la consola de AWS Glue. Cuando este trabajo se ejecuta, también se debe proporcionar un rol de IAM con los permisos correspondientes.

Los valores de parámetro comunes se muestran en la plantilla. Por ejemplo, el valor predeterminado de `AllocatedCapacity` (DPU) es 5.

```
---
AWSTemplateFormatVersion: '2010-09-09'
# Sample CFN YAML to demonstrate creating a job using a MySQL JDBC DB with the flights
 data to an S3 file
#
# Parameters section contains names that are substituted in the Resources section
# These parameters are the names the resources created in the Data Catalog
Parameters:

# The name of the job to be created
  CFNJobName:
    Type: String
    Default: cfn-job-JDBC-to-S3-1
# The name of the IAM role that the job assumes. It must have access to data, script,
 temporary directory
  CFNIAMRoleName:
    Type: String
    Default: AWSGlueServiceRoleGA
# The S3 path where the script for this job is located
  CFNScriptLocation:
    Type: String
    Default: s3://aws-glue-scripts-123456789012-us-east-1/myid/sal-job-dec4a
# The name of the connection used for JDBC data source
  CFNConnectionName:
    Type: String
    Default: cfn-connection-mysql-flights-1
#
#
# Resources section defines metadata for the Data Catalog
Resources:
# Create job to run script which accesses JDBC flights table via a connection and write
 to S3 file as parquet.
# The script already exists and is called by this job
  CFNJobFlights:
    Type: AWS::Glue::Job
    Properties:
      Role: !Ref CFNIAMRoleName
      #DefaultArguments: JSON object
```

```
      # For example, if required by script, set temporary directory as
 DefaultArguments={'--TempDir'; 's3://aws-glue-temporary-xyc/sal'}
      Connections:
        Connections:
        - !Ref CFNConnectionName
      #MaxRetries: Double
      Description: Job created with CloudFormation using existing script
      #LogUri: String
      Command:
        Name: glueetl
        ScriptLocation: !Ref CFNScriptLocation
             # for access to directories use proper IAM role with permission to buckets
 and folders that begin with "aws-glue-"
             # if required, script defines temp directory as argument TempDir and used
 in script like redshift_tmp_dir = args["TempDir"]
             # script defines target for output as s3://aws-glue-target/sal
      AllocatedCapacity: 5
      ExecutionProperty:
        MaxConcurrentRuns: 1
      Name: !Ref CFNJobName
```

Ejemplo de plantilla de AWS CloudFormation para un desencadenador bajo demanda de AWS Glue

Un desencadenador de AWS Glue en el Catálogo de datos contiene los valores de parámetros que son necesarios para iniciar una ejecución de flujo de trabajo cuando se activa el desencadenador. Un disparador bajo demanda se activa cuando lo habilita.

Esta muestra crea un disparador bajo demanda que comienza un flujo de trabajo llamado `cfn-job-S3-to-S3-1`.

```
---
AWSTemplateFormatVersion: '2010-09-09'
# Sample CFN YAML to demonstrate creating an on-demand trigger
#
# Parameters section contains names that are substituted in the Resources section
# These parameters are the names the resources created in the Data Catalog
Parameters:
  # The existing job to be started by this trigger
  CFNJobName:
```

```
    Type: String
    Default: cfn-job-S3-to-S3-1
  # The name of the trigger to be created
  CFNTriggerName:
    Type: String
    Default: cfn-trigger-ondemand-flights-1
#
# Resources section defines metadata for the Data Catalog
# Sample CFN YAML to demonstrate creating an on-demand trigger for a job
Resources:
# Create trigger to run an existing job (CFNJobName) on an on-demand schedule.
  CFNTriggerSample:
    Type: AWS::Glue::Trigger
    Properties:
      Name:
        Ref: CFNTriggerName
      Description: Trigger created with CloudFormation
      Type: ON_DEMAND
      Actions:
        - JobName: !Ref CFNJobName
        # Arguments: JSON object
      #Schedule:
      #Predicate:
```

Plantilla de AWS CloudFormation de muestra para un disparador programado de AWS Glue

Un desencadenador de AWS Glue en el Catálogo de datos contiene los valores de parámetros que son necesarios para iniciar una ejecución de flujo de trabajo cuando se activa el desencadenador. Un disparador programado se activa cuando está habilitado y se activa el temporizador cron.

Esta muestra crea un disparador programado que comienza un flujo de trabajo llamado `cfn-job-S3-to-S3-1`. El temporizador es una expresión cron para ejecutar el flujo de trabajo cada 10 minutos los días entre semana.

```
---
AWSTemplateFormatVersion: '2010-09-09'
# Sample CFN YAML to demonstrate creating a scheduled trigger
#
# Parameters section contains names that are substituted in the Resources section
```

```
# These parameters are the names the resources created in the Data Catalog
Parameters:
  # The existing job to be started by this trigger
  CFNJobName:
    Type: String
    Default: cfn-job-S3-to-S3-1
  # The name of the trigger to be created
  CFNTriggerName:
    Type: String
    Default: cfn-trigger-scheduled-flights-1
#
# Resources section defines metadata for the Data Catalog
# Sample CFN YAML to demonstrate creating a scheduled trigger for a job
#
Resources:
# Create trigger to run an existing job (CFNJobName) on a cron schedule.
  TriggerSample1CFN:
    Type: AWS::Glue::Trigger
    Properties:
      Name:
        Ref: CFNTriggerName
      Description: Trigger created with CloudFormation
      Type: SCHEDULED
      Actions:
        - JobName: !Ref CFNJobName
        # Arguments: JSON object
      # # Run the trigger every 10 minutes on Monday to Friday
      Schedule: cron(0/10 * ? * MON-FRI *)
      #Predicate:
```

Plantilla de AWS CloudFormation de muestra para un disparador condicional de AWS Glue

Un desencadenador de AWS Glue en el Catálogo de datos contiene los valores de parámetros que son necesarios para iniciar una ejecución de flujo de trabajo cuando se activa el desencadenador. Un disparador condicional se activa cuando está habilitado y se cumplen sus condiciones, por ejemplo, que un flujo de trabajo se realice correctamente.

Esta muestra crea un disparador condicional que comienza un flujo de trabajo llamado `cfn-job-S3-to-S3-1`. Este flujo de trabajo comienza cuando el flujo de trabajo llamado `cfn-job-S3-to-S3-2` se completa correctamente.

```
---
AWSTemplateFormatVersion: '2010-09-09'
# Sample CFN YAML to demonstrate creating a conditional trigger for a job, which starts
 when another job completes
#
# Parameters section contains names that are substituted in the Resources section
# These parameters are the names the resources created in the Data Catalog
Parameters:
  # The existing job to be started by this trigger
  CFNJobName:
    Type: String
    Default: cfn-job-S3-to-S3-1
  # The existing job that when it finishes causes trigger to fire
  CFNJobName2:
    Type: String
    Default: cfn-job-S3-to-S3-2
  # The name of the trigger to be created
  CFNTriggerName:
    Type: String
    Default: cfn-trigger-conditional-1
#
Resources:
# Create trigger to run an existing job (CFNJobName) when another job completes
 (CFNJobName2).
  CFNTriggerSample:
    Type: AWS::Glue::Trigger
    Properties:
      Name:
        Ref: CFNTriggerName
      Description: Trigger created with CloudFormation
      Type: CONDITIONAL
      Actions:
        - JobName: !Ref CFNJobName
        # Arguments: JSON object
      #Schedule: none
      Predicate:
        #Value for Logical is required if more than 1 job listed in Conditions
        Logical: AND
        Conditions:
          - LogicalOperator: EQUALS
            JobName: !Ref CFNJobName2
            State: SUCCEEDED
```

Plantilla de AWS CloudFormation de muestra para un punto de conexión de desarrollo de AWS Glue

Una transformación de machine learning de AWS Glue es una transformación personalizada para limpiar los datos. En la actualidad, hay una transformación disponible denominada FindMatches. La transformación FindMatches le permite identificar registros duplicados o coincidentes en el conjunto de datos, incluso cuando los registros no tienen un identificador único común y no coinciden exactamente los campos.

Este ejemplo crea una transformación de machine learning. Para obtener más información sobre los parámetros que se necesitan para crear una transformación de machine learning, consulte Coincidencia de registros con FindMatches de AWS Lake Formation.

```
---
AWSTemplateFormatVersion: '2010-09-09'
# Sample CFN YAML to demonstrate creating a machine learning transform
#
# Resources section defines metadata for the machine learning transform
Resources:
  MyMLTransform:
    Type: "AWS::Glue::MLTransform"
    Condition: "isGlueMLGARegion"
    Properties:
      Name: !Sub "MyTransform"
      Description: "The bestest transform ever"
      Role: !ImportValue MyMLTransformUserRole
      GlueVersion: "1.0"
      WorkerType: "Standard"
      NumberOfWorkers: 5
      Timeout: 120
      MaxRetries: 1
      InputRecordTables:
        GlueTables:
          - DatabaseName: !ImportValue MyMLTransformDatabase
            TableName: !ImportValue MyMLTransformTable
      TransformParameters:
        TransformType: "FIND_MATCHES"
        FindMatchesParameters:
          PrimaryKeyColumnName: "testcolumn"
          PrecisionRecallTradeoff: 0.5
```

```
          AccuracyCostTradeoff: 0.5
          EnforceProvidedLabels: True
      Tags:
        key1: "value1"
        key2: "value2"
      TransformEncryption:
        TaskRunSecurityConfigurationName: !ImportValue
 MyMLTransformSecurityConfiguration
        MLUserDataEncryption:
          MLUserDataEncryptionMode: "SSE-KMS"
          KmsKeyId: !ImportValue MyMLTransformEncryptionKey
```

Ejemplo de plantilla de AWS CloudFormation para un conjunto de reglas de AWS Glue Data Quality

Un conjunto de reglas de calidad de datos de AWS Glue contiene reglas que se pueden evaluar en una tabla del catálogo de datos. Una vez colocado el conjunto de reglas en la tabla de destino, puede ir al catálogo de datos y realizar una evaluación para comparar los datos con las reglas del conjunto de reglas. Estas reglas pueden variar desde la evaluación del recuento de filas hasta la evaluación de la integridad referencial de los datos.

En el siguiente ejemplo, se muestra una plantilla de CloudFormation que crea un conjunto de reglas con una variedad de reglas en la tabla de destino especificada.

```
AWSTemplateFormatVersion: '2010-09-09'
# Sample CFN YAML to demonstrate creating a DataQualityRuleset
#
# Parameters section contains names that are substituted in the Resources section
# These parameters are the names the resources created in the Data Catalog
Parameters:

  # The name of the ruleset to be created
  RulesetName:
    Type: String
    Default: "CFNRulesetName"
  RulesetDescription:
    Type: String
    Default: "CFN DataQualityRuleset"
  # Rules that will be associated with this ruleset
  Rules:
    Type: String
```

```
    Default: 'Rules = [
        RowCount > 100,
        IsUnique "id",
        IsComplete "nametype"
        ]'
  # Name of database and table within Data Catalog which the ruleset will
  # be applied too
  DatabaseName:
    Type: String
    Default: "ExampleDatabaseName"
  TableName:
    Type: String
    Default: "ExampleTableName"

# Resources section defines metadata for the Data Catalog
Resources:
  # Creates a Data Quality ruleset under specified rules
  DQRuleset:
    Type: AWS::Glue::DataQualityRuleset
    Properties:
      Name: !Ref RulesetName
      Description: !Ref RulesetDescription
      # The String within rules must be formatted in DQDL, a language
      # used specifically to make rules
      Ruleset: !Ref Rules
      # The targeted table must exist within Data Catalog alongside
      # the correct database
      TargetTable:
        DatabaseName: !Ref DatabaseName
        TableName: !Ref TableName
```

Plantilla de AWS CloudFormation de ejemplo para un conjunto de reglas de AWS Glue Data Quality con el programador de EventBridge

Un conjunto de reglas de calidad de datos de AWS Glue contiene reglas que se pueden evaluar en una tabla del catálogo de datos. Una vez colocado el conjunto de reglas en la tabla de destino, puede ir al catálogo de datos y realizar una evaluación para comparar los datos con las reglas del conjunto de reglas. En lugar de tener que ir manualmente al catálogo de datos para evaluar el conjunto de reglas, también puede agregar un EventBridge Scheduler a nuestra plantilla de

CloudFormation para programar estas evaluaciones del conjunto de reglas en un intervalo de tiempo determinado.

El siguiente ejemplo es una plantilla de CloudFormation que crea un conjunto de reglas de calidad de datos y un programador de EventBridge para evaluar el conjunto de reglas mencionado anteriormente cada cinco minutos.

```
AWSTemplateFormatVersion: '2010-09-09'
# Sample CFN YAML to demonstrate creating a DataQualityRuleset
#
# Parameters section contains names that are substituted in the Resources section
# These parameters are the names the resources created in the Data Catalog
Parameters:

  # The name of the ruleset to be created
  RulesetName:
    Type: String
    Default: "CFNRulesetName"
  # Rules that will be associated with this Ruleset
  Rules:
    Type: String
    Default: 'Rules = [
        RowCount > 100,
        IsUnique "id",
        IsComplete "nametype"
        ]'
  # The name of the Schedule to be created
  ScheduleName:
    Type: String
    Default: "ScheduleDQRulsetEvaluation"
  # This expression determines the rate at which the Schedule will evaluate
  # your data using the above ruleset
  ScheduleRate:
    Type: String
    Default: "rate(5 minutes)"
  # The Request that being sent must match the details of the Data Quality Ruleset
  ScheduleRequest:
    Type: String
    Default: '
        { "DataSource": { "GlueTable": { "DatabaseName": "ExampleDatabaseName",
         "TableName": "ExampleTableName" } },
         "Role": "role/AWSGlueServiceRoleDefault",
          "RulesetNames": [ ""CFNRulesetName"" ] }
```

```
        '

# Resources section defines metadata for the Data Catalog
Resources:
  # Creates a Data Quality ruleset under specified rules
  DQRuleset:
    Type: AWS::Glue::DataQualityRuleset
    Properties:
      Name: !Ref RulesetName
      Description: "CFN DataQualityRuleset"
      # The String within rules must be formatted in DQDL, a language
      # used specifically to make rules
      Ruleset: !Ref Rules
      # The targeted table must exist within Data Catalog alongside
      # the correct database
      TargetTable:
        DatabaseName: "ExampleDatabaseName"
        TableName: "ExampleTableName"
  # Create a Scheduler to schedule evaluation runs on the above ruleset
  ScheduleDQEval:
    Type: AWS::Scheduler::Schedule
    Properties:
      Name: !Ref ScheduleName
      Description: "Schedule DataQualityRuleset Evaluations"
      FlexibleTimeWindow:
        Mode: "OFF"
      ScheduleExpression: !Ref ScheduleRate
      ScheduleExpressionTimezone: "America/New_York"
      State: "ENABLED"
      Target:
        # The ARN is the API that will be run, since we want to evaluate our ruleset
        # we want this specific ARN
        Arn: "arn:aws:scheduler:::aws-sdk:glue:startDataQualityRulesetEvaluationRun"
        # Your RoleArn must have approval to schedule
        RoleArn: "arn:aws:iam::123456789012:role/AWSGlueServiceRoleDefault"
        # This is the Request that is being sent to the Arn
        Input: '
        { "DataSource": { "GlueTable": { "DatabaseName": "sampledb", "TableName":
 "meteorite" } },
         "Role": "role/AWSGlueServiceRoleDefault",
          "RulesetNames": [ "TestCFN" ] }
        '
```

Plantilla de AWS CloudFormation de muestra para un punto de conexión de desarrollo de AWS Glue

Un punto de enlace de desarrollo de AWS Glue es un entorno que puede utilizar para desarrollar y probar los scripts de AWS Glue.

Esta muestra crea un punto de enlace de desarrollo con los valores de parámetro de red mínimos necesarios para crearlo correctamente. Para obtener más información acerca de los parámetros que necesita para configurar un punto de enlace de desarrollo, consulte Configuración de redes para el desarrollo de AWS Glue.

Proporcione un ARN (nombre de recurso de Amazon) de rol de IAM existente para crear el punto de enlace de desarrollo. Proporcione una clave pública RSA válida y mantenga disponible la clave privada correspondiente si tiene previsto crear un servidor de blocs de notas en el punto de enlace de desarrollo.

> **Note**
>
> Usted administrará los servidores de blocs de notas que cree y estén asociados con un punto de enlace de desarrollo. Por tanto, si elimina el punto de enlace de desarrollo, para eliminar el bloc de notas, será preciso que elimine la pila de AWS CloudFormation en la consola de AWS CloudFormation.

```
---
AWSTemplateFormatVersion: '2010-09-09'
# Sample CFN YAML to demonstrate creating a development endpoint
#
# Parameters section contains names that are substituted in the Resources section
# These parameters are the names the resources created in the Data Catalog
Parameters:

# The name of the crawler to be created
  CFNEndpointName:
    Type: String
    Default: cfn-devendpoint-1
  CFNIAMRoleArn:
    Type: String
```

```
    Default: arn:aws:iam::123456789012/role/AWSGlueServiceRoleGA
#
#
# Resources section defines metadata for the Data Catalog
Resources:
  CFNDevEndpoint:
    Type: AWS::Glue::DevEndpoint
    Properties:
      EndpointName: !Ref CFNEndpointName
      #ExtraJarsS3Path: String
      #ExtraPythonLibsS3Path: String
      NumberOfNodes: 5
      PublicKey: ssh-rsa public.....key myuserid-key
      RoleArn: !Ref CFNIAMRoleArn
      SecurityGroupIds:
        - sg-64986c0b
      SubnetId: subnet-c67cccac
```

Guía de programación de AWS Glue

Los scripts contienen código que extrae datos de orígenes, los transforma y los carga en destinos. AWS Glue ejecuta un script cuando inicia un flujo de trabajo.

Los scripts de ETL de AWS Glue están codificados en Python o Scala. Si bien todos los tipos de trabajo se pueden escribir en Python, en el AWS Glue caso de Spark, los trabajos también se pueden escribir en Scala. Al generar automáticamente la lógica de código de origen para el trabajo en AWS Glue Studio, se crea un script. Puede editar este script o proporcionar su propio script para procesar el flujo de trabajo de ETL.

Suministro de sus propios scripts personalizados

Los scripts realizan el flujo de trabajo de extracción, transformación y carga (ETL) en AWS Glue. Un script se crea al generar automáticamente lógica de código fuente para un flujo de trabajo. Puede editar este script generado o proporcionar su propio script personalizado.

Para proporcionar su propio script personalizado en AWS Glue, siga estos pasos generales:

1. Inicie sesión en la AWS Management Console y abra la consola de AWS Glue en https://console.aws.amazon.com/glue/.
2. Seleccione la pestaña Trabajos de ETL y, luego, consulte la sección Crear trabajo. Elija una opción de editor de script.
3. En This job runs (Ejecuciones de este flujo de trabajo), elija una de las siguientes opciones:
   - Crear un script nuevo con código repetitivo
   - Cargar y editar un script existente
4. En la página Propiedades de trabajo, elija rol de IAM necesaria para la ejecución de su script personalizado. Para obtener más información, consulte Gestión de identidad y acceso para AWS Glue.
5. Elija cualquier conexión a la que su script haga referencia. Estos objetos son necesarios para conectarse a los almacenes de datos de JDBC necesarios.

   Una interfaz de red elástica es una interfaz de red virtual que puede asociar a una instancia en una nube virtual privada (VPC). Elija la interfaz de red elástica necesaria para conectarse al almacén de datos que se usa en el script.

6. Proporcione una configuración adicional, incluidos los parámetros, específica para su tipo de trabajo. Para obtener más información sobre la configuración de su tipo de trabajo, consulte la sección Creación de trabajos de ETL visuales con AWS Glue Studio.
7. En la pestaña Script, pegue o escriba su script personalizado.

Utilice el contenido de esta sección para guiar el proceso de escritura de su guion personalizado.

Para obtener más información acerca de la adición de trabajos en AWS Glue, consulte Creación de trabajos de ETL visuales con AWS Glue Studio.

Para obtener instrucciones paso a paso, consulte el tutorial Add job (Añadir flujo de trabajo) en la consola de AWS Glue.

Programación de scripts de Spark

Con AWS Glue resulta más sencillo escribir o autogenerar scripts de extracción, transformación y carga (ETL), además de probarlos y ejecutarlos. En esta sección se describen las extensiones de Apache Spark que se han introducido con AWS Glue y se ofrecen ejemplos sobre cómo codificar y ejecutar scripts de ETL en Python y Scala.

> **Important**
>
> Las distintas versiones de AWS Glue admiten diferentes versiones de Apache Spark. Su script personalizado debe ser compatible con la versión de Apache Spark admitida. Para obtener información acerca de las versiones de AWS Glue, consulte Glue version job property.

Temas

- Tutorial: Cómo escribir un guion de AWS Glue for Spark
- Programe scripts AWS Glue ETL en PySpark
- Programación de scripts de ETL de AWS Glue en Scala
- Características y optimizaciones para programar AWS Glue para los scripts de Spark ETL

Tutorial: Cómo escribir un guion de AWS Glue for Spark

Este tutorial te presenta el proceso de escribir guiones de AWS Glue. Puede ejecutar scripts de manera programada con trabajos, o bien de manera interactiva con sesiones interactivas. Para obtener más información acerca de los trabajos, consulte Creación de trabajos de ETL visuales con AWS Glue Studio. Para más información sobre las sesiones interactivas, consulte the section called "Información general sobre las sesiones interactivas de AWS Glue".

El editor visual de AWS Glue Studio ofrece una interfaz gráfica sin código para crear trabajos de AWS Glue. AWS Glue reproduce los guiones de los trabajos visuales. Le dan acceso al conjunto ampliado de herramientas disponibles para trabajar con los programas de Apache Spark. Puedes acceder a las API nativas de Spark, así como a las bibliotecas de AWS Glue que facilitan los flujos de trabajo de extracción, transformación y carga (ETL) desde un script de AWS Glue.

En este tutorial, extraerá, transformará y cargará un conjunto de datos de multas de estacionamiento. El script que realiza este trabajo es idéntico en forma y función al generado en Making ETL easy with AWS Glue Studio en el blog AWS Big Data, que presenta el editor visual AWS Glue Studio. Al ejecutar este script en un trabajo, puede compararlo con trabajos visuales y ver cómo funcionan los scripts ETL de AWS Glue. Esto le permite usar funcionalidades adicionales que aún no están disponibles en los trabajos visuales.

En este tutorial utilizará lenguaje y bibliotecas Python. Scala dispone de funcionalidad similar. Después de seguir este tutorial, debería poder generar e inspeccionar un ejemplo de script de Scala para comprender cómo realizar el proceso de escritura de scripts ETL de Scala AWS Glue.

Requisitos previos

Este tutorial tiene los requisitos previos siguientes:

- Los mismos requisitos previos que en la entrada del blog de AWS Glue Studio, en la que se indica cómo ejecutar una AWS CloudFormation plantilla.

  Esta plantilla usa el catálogo de datos de AWS Glue para administrar el conjunto de datos de multas de estacionamiento disponible en `s3://aws-bigdata-blog/artifacts/gluestudio/`. Crea los siguientes recursos a los que se hará referencia:
- AWS Glue StudioRole (Rol): rol de IAM que ejecuta trabajos de AWS Glue
- AWS Glue StudioAmazon S3 Bucket (Bucket de Amazon S3): nombre del bucket de Amazon S3 para almacenar archivos relacionados con el blog
- AWS Glue StudioTicketsYYZDB: base de datos de AWS Glue Data Catalog

- AWS Glue StudioTableTickets— Tabla del catálogo de datos para usar como fuente
- AWS Glue StudioTableTrials— Tabla del catálogo de datos para usar como fuente
- AWS Glue StudioParkingTicketCount — Tabla del catálogo de datos para usar como destino
- El script generado en la entrada del blog de AWS Glue Studio. En caso de que la publicación se modifique, el script también está disponible en el siguiente texto.

Generación de un script de ejemplo

Puede utilizar el editor visual de AWS Glue Studio como una potente herramienta de generación de código para crear un andamiaje para el guion que desee escribir. Utilizará esta herramienta para crear un script de ejemplo.

Si quiere omitir estos pasos, se proporciona el script.

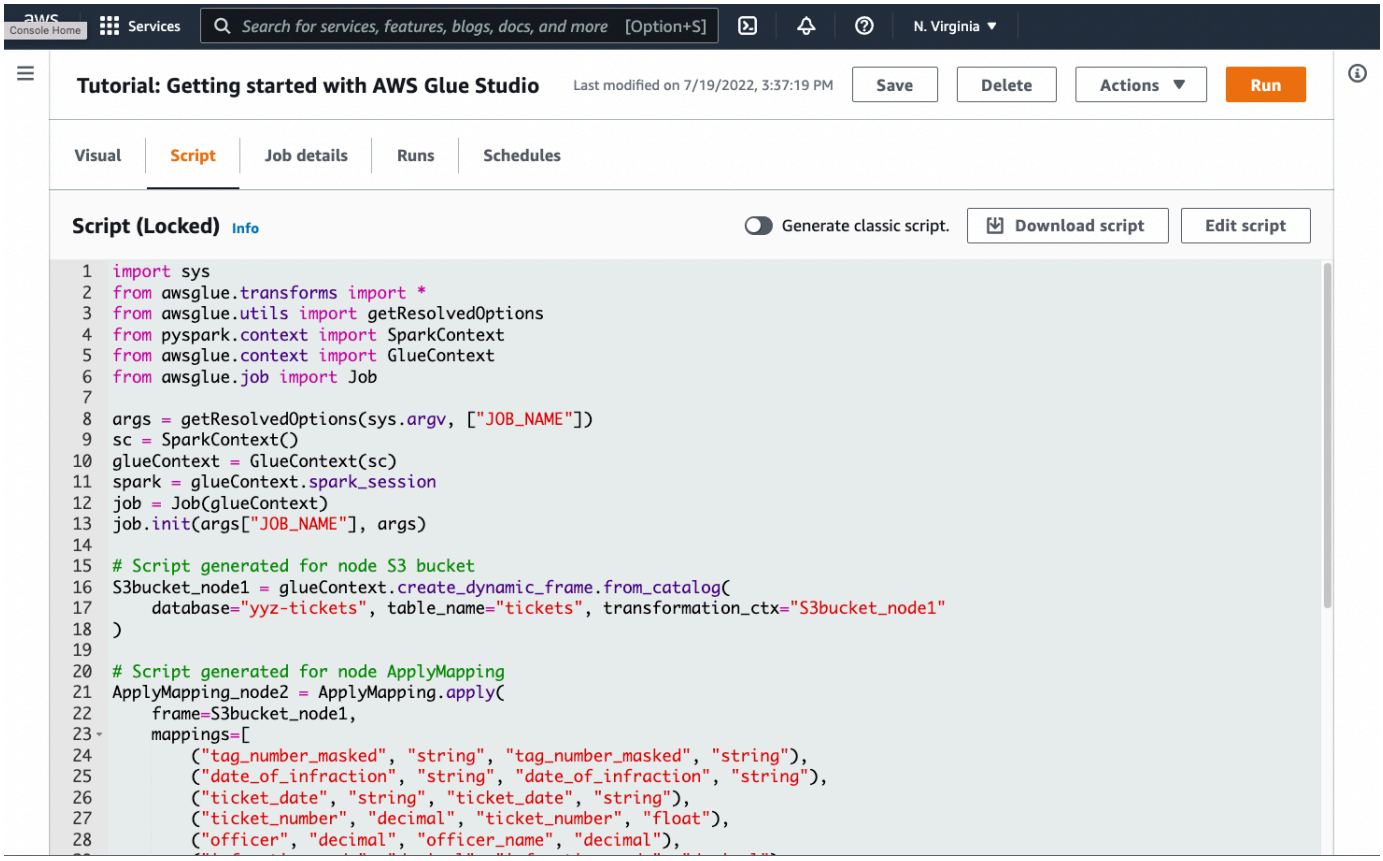
Script de ejemplo del tutorial

```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job

args = getResolvedOptions(sys.argv, ["JOB_NAME"])
sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args["JOB_NAME"], args)

# Script generated for node S3 bucket
S3bucket_node1 = glueContext.create_dynamic_frame.from_catalog(
    database="yyz-tickets", table_name="tickets", transformation_ctx="S3bucket_node1"
)

# Script generated for node ApplyMapping
ApplyMapping_node2 = ApplyMapping.apply(
    frame=S3bucket_node1,
    mappings=[
        ("tag_number_masked", "string", "tag_number_masked", "string"),
        ("date_of_infraction", "string", "date_of_infraction", "string"),
```

```
        ("ticket_date", "string", "ticket_date", "string"),
        ("ticket_number", "decimal", "ticket_number", "float"),
        ("officer", "decimal", "officer_name", "decimal"),
        ("infraction_code", "decimal", "infraction_code", "decimal"),
        ("infraction_description", "string", "infraction_description", "string"),
        ("set_fine_amount", "decimal", "set_fine_amount", "float"),
        ("time_of_infraction", "decimal", "time_of_infraction", "decimal"),
    ],
    transformation_ctx="ApplyMapping_node2",
)

# Script generated for node S3 bucket
S3bucket_node3 = glueContext.write_dynamic_frame.from_options(
    frame=ApplyMapping_node2,
    connection_type="s3",
    format="glueparquet",
    connection_options={"path": "s3://DOC-EXAMPLE-BUCKET", "partitionKeys": []},
    format_options={"compression": "gzip"},
    transformation_ctx="S3bucket_node3",
)

job.commit()
```

Para generar un script de ejemplo

1. Completa el tutorial de AWS Glue Studio. Para completar este tutorial, consulte Creación de un trabajo en AWS Glue Studio a partir de un trabajo de ejemplo.
2. Vaya a la pestaña Script de la página de trabajo, tal como se muestra en la siguiente captura de pantalla:



3. Copie el contenido completo de la pestaña Script. Configurando el lenguaje en Job details (Detalles del trabajo), puede cambiar y alternar entre generar código Python o Scala.

Paso 1. Crear un trabajo y pegar el script

En este paso, crearás un trabajo de AWS Glue en el AWS Management Console. Esto establece una configuración que permite a AWS Glue ejecutar el script. Además, crea un lugar donde almacenar y editar el script.

Creación de un trabajo

1. En AWS Management Console, navega hasta la página de inicio de AWS Glue.
2. En el panel de navegación lateral, seleccione Trabajos.
3. Elija Editor de scripts de Spark en Crear trabajo, a continuación, elija Crear.
4. Opcional Pegue todo el texto del script en el panel Script. Como alternativa, puede seguir el tutorial.

Paso 2. Importación de bibliotecas de AWS Glue

Es necesario configurar el script para que interactúe con el código y la configuración que se definen fuera del script. Este trabajo se realiza entre bastidores en AWS Glue Studio.

En este tutorial, se realizan las siguientes acciones.

- Importar e inicializar un objeto de `GlueContext`. Esta es la importación más importante, desde la perspectiva de la escritura de scripts. Esto expone métodos estándar para definir los conjuntos de datos de origen y de destino, que es el punto de partida de cualquier script ETL. Para obtener más información sobre la clase `GlueContext`, consulte GlueContext clase.
- Inicializar una clase `SparkContext` y una clase `SparkSession`. Estas le permiten configurar el motor Spark disponible dentro de la tarea de AWS Glue. No necesitarás usarlos directamente en los guiones introductorios de AWS Glue.
- Llamar a `getResolvedOptions` con el fin de preparar los argumentos del trabajo para su uso en el script. Para obtener más información sobre la resolución de parámetros de trabajo, consulte the section called “getResolvedOptions”.
- Inicializar un objeto `Job`. El `Job` objeto establece la configuración y realiza un seguimiento del estado de varias funciones opcionales de AWS Glue. El script puede ejecutarse sin un objeto `Job`, pero lo mejor es inicializarlo para que no se produzcan confusiones si esas funciones se integran posteriormente.

  Una de esas características son los marcadores de trabajo, que puede configurar de manera opcional en este tutorial. Puede obtener información sobre los marcadores de trabajo en la siguiente sección, the section called “Opcional: Habilitar marcadores de trabajo”.

En este procedimiento, escribirá el siguiente código. Este código es una parte del script de ejemplo generado.

```
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job

args = getResolvedOptions(sys.argv, ["JOB_NAME"])
sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
```

```
job = Job(glueContext)
job.init(args["JOB_NAME"], args)
```

Para importar bibliotecas de AWS Glue

- Copie esta sección de código y péguela en el editor de Script.

> **Note**
>
> Es posible que copiar código le parezca una mala práctica de ingeniería. En este tutorial, te sugerimos esto para animarte a nombrar tus variables principales de forma coherente en todos los scripts ETL de AWS Glue.

Paso 3. Extraer datos de un origen

En cualquier proceso de ETL, primero se debe definir el conjunto de datos de origen que quiere cambiar. En el editor visual de AWS Glue Studio, se proporciona esta información mediante la creación de un nodo Source.

En este paso, se proporciona el método `create_dynamic_frame.from_catalog`, un `database` y `table_name`, para extraer datos de un origen configurado en el Catálogo de datos de AWS Glue.

En el paso anterior, se inicializó un objeto `GlueContext`. Ese objeto se emplea para buscar métodos que se utilizan para configurar orígenes, como `create_dynamic_frame.from_catalog`.

En este procedimiento, escribirá el siguiente código utilizando `create_dynamic_frame.from_catalog`. Este código es una parte del script de ejemplo generado.

```
S3bucket_node1 = glueContext.create_dynamic_frame.from_catalog(
    database="yyz-tickets", table_name="tickets", transformation_ctx="S3bucket_node1"
    )
```

Para extraer datos de un origen

1. Examine la documentación para encontrar un método `GlueContext` para extraer datos de una fuente definida en el catálogo de datos de AWS Glue. Estos métodos están documentados en

the section called “GlueContext”. Elija el método create_dynamic_frame.from_catalog. Llame a este método en `glueContext`.

2. Examine la documentación de `create_dynamic_frame.from_catalog`. Este método necesita los parámetros `database` y `table_name`. Proporcione los parámetros necesarios a `create_dynamic_frame.from_catalog`.

   El catálogo de datos de AWS Glue almacena información sobre la ubicación y el formato de los datos de origen y se configuró en la sección de requisitos previos. No es necesario proporcionar directamente esa información al script.

3. Opcional: proporcione el parámetro `transformation_ctx` al método para que se admitan marcadores de trabajo. Puede obtener información sobre los marcadores de trabajo en la siguiente sección, the section called “Opcional: Habilitar marcadores de trabajo”.

> **Note**
>
> Métodos habituales para extraer datos
> the section called “create_dynamic_frame_from_catalog”se utiliza para conectarse a las tablas del catálogo de datos de AWS Glue.
> Si necesita proporcionar directamente al trabajo una configuración que describa la estructura y ubicación del origen, consulte el método the section called “create_dynamic_frame_from_options”. Deberá proporcionar parámetros más detallados que describan los datos que si se utiliza `create_dynamic_frame.from_catalog`. Consulte la documentación suplementaria sobre `format_options` y `connection_parameters` para identificar los parámetros necesarios. Para ver una explicación sobre cómo proporcionar al script información sobre el formato de los datos de origen, consulte the section called “Opciones de formato de datos”. Para ver una explicación sobre cómo proporcionar al script información sobre la ubicación de los datos de origen, consulte the section called “Parámetros de conexión”.
> Si va a leer información de un origen que transmite flujos, debe proporcionar al trabajo información sobre el origen a través de los métodos the section called “create_data_frame_from_catalog” o the section called “create_data_frame_from_options”.
> Tenga en cuenta que estos métodos devuelven `DataFrames` de Apache Spark.
> Nuestro código generado llama a `create_dynamic_frame.from_catalog`, mientras que en la documentación de referencia se alude a `create_dynamic_frame_from_catalog`.
> En última instancia, estos métodos llaman al mismo código, y se incluyen para que pueda

escribir un código más limpio. Puede comprobar esto visualizando el código de origen de nuestro encapsulador de Python, disponible en aws-glue-libs.

Paso 4. Transformar los datos con AWS Glue

Después de extraer los datos de origen en un proceso de ETL, debe describir cómo desea cambiar los datos. Esta información se proporciona mediante la creación de un nodo Transform en el editor visual de AWS Glue Studio.

En este paso, debe proporcionarle al método `ApplyMapping` un mapa de los nombres y tipos de campos actuales y deseados para transformar `DynamicFrame`.

Se deben realizar las siguientes transformaciones.

- Eliminar las cuatro claves `location` y `province`.
- Cambiar el nombre de `officer` por `officer_name`.
- Cambiar el tipo de `ticket_number` y `set_fine_amount` por `float`.

`create_dynamic_frame.from_catalog` proporciona un objeto `DynamicFrame`. A `DynamicFrame` representa un conjunto de datos en AWS Glue. AWS Las transformaciones de Glue son operaciones que cambian`DynamicFrames`.

> **Note**
>
> ¿Qué es un `DynamicFrame`?
> Un `DynamicFrame` es una abstracción que permite conectar un conjunto de datos con una descripción de los nombres y tipos de entradas de los datos. En Apache Spark, existe una abstracción similar llamada a DataFrame. Para obtener una explicación DataFrames, consulte la Guía SQL de Spark.
> Con `DynamicFrames`, puede describir los esquemas de conjuntos de datos de forma dinámica. Considere un conjunto de datos con una columna de precios, donde algunas entradas almacenan el precio como una cadena y otras almacenan el precio como un doble. AWS Glue calcula un esquema on-the-fly: crea un registro autodescriptivo para cada fila. Los campos incoherentes (como el precio) se representan explícitamente con un tipo (`ChoiceType`) en el esquema del marco. Los campos incoherentes se pueden eliminar con `DropFields`, o bien resolverlos con `ResolveChoice`. Estas son transformaciones que

están disponibles en el `DynamicFrame`. Después, los datos se pueden volver a escribir en el lago de datos con `writeDynamicFrame`.
Puede llamar a muchas de las mismas transformaciones desde métodos de la clase `DynamicFrame`, lo que puede dar como resultado scripts más legibles. Para obtener más información acerca de `DynamicFrame`, consulte the section called “DynamicFrame”.

En este procedimiento, escribirá el siguiente código utilizando `ApplyMapping`. Este código es una parte del script de ejemplo generado.

```
ApplyMapping_node2 = ApplyMapping.apply(
    frame=S3bucket_node1,
    mappings=[
        ("tag_number_masked", "string", "tag_number_masked", "string"),
        ("date_of_infraction", "string", "date_of_infraction", "string"),
        ("ticket_date", "string", "ticket_date", "string"),
        ("ticket_number", "decimal", "ticket_number", "float"),
        ("officer", "decimal", "officer_name", "decimal"),
        ("infraction_code", "decimal", "infraction_code", "decimal"),
        ("infraction_description", "string", "infraction_description", "string"),
        ("set_fine_amount", "decimal", "set_fine_amount", "float"),
        ("time_of_infraction", "decimal", "time_of_infraction", "decimal"),
    ],
    transformation_ctx="ApplyMapping_node2",
)
```

Para transformar datos con AWS Glue

1. Examine la documentación para identificar una transformación para cambiar y eliminar campos. Para obtener más detalles, consulte the section called “GlueTransform”. Elija la transformación `ApplyMapping`. Para obtener más información acerca de `ApplyMapping`, consulte the section called “ApplyMapping”. Llame a `apply` en el objeto de transformación `ApplyMapping`.

   > **Note**
   >
   > ¿Qué es `ApplyMapping`?
   > `ApplyMapping` toma un `DynamicFrame` y lo transforma. Toma una lista de tuplas que representan transformaciones en los campos, una "asignación". Los dos primeros elementos de la tupla, un nombre y un tipo de campo, se utilizan para identificar un

campo en el marco. Los dos segundos parámetros también son un nombre y un tipo de campo.
ApplyMapping convierte el campo de origen en el nombre de destino y escribe uno nuevo`DynamicFrame`, que devuelve. Los campos que no se proporcionan se eliminan en el valor devuelto.
En lugar de llamar a `apply`, se puede llamar a la misma transformación con el método `apply_mapping` de `DynamicFrame` para crear código más fluido y legible. Para obtener más información, consulte the section called “apply_mapping”.

2. Examine la documentación de `ApplyMapping` para identificar los parámetros requeridos. Consulte the section called “ApplyMapping”. Descubrirá que este método requiere los parámetros `frame` y `mappings`. Proporcione los parámetros necesarios a `ApplyMapping`.
3. Opcional: proporcione `transformation_ctx` al método para que se admitan marcadores de trabajo. Puede obtener información sobre los marcadores de trabajo en la siguiente sección, the section called “Opcional: Habilitar marcadores de trabajo”.

> **Note**
>
> Funcionalidad de Apache Spark
> Proporcionamos transformaciones para agilizar los flujos de trabajo de ETL dentro de un trabajo. Además, también puede acceder a las bibliotecas disponibles en un programa de Spark en el trabajo, creadas con fines más generales. Para utilizarlas, debe realizar conversiones entre `DynamicFrame` y `DataFrame`.
> Puede crear `DataFrame` con the section called “toDF”. A continuación, puede utilizar los métodos disponibles en el DataFrame para transformar el conjunto de datos. Para obtener más información sobre estos métodos, consulte DataFrame. A continuación, puede realizar la conversión al revés the section called “fromDF” para utilizar las operaciones de AWS Glue para cargar el fotograma en un objetivo.

Paso 5. Cargar datos en un destino

Después de transformar los datos, normalmente se almacenan los datos transformados en un lugar distinto al origen. Para realizar esta operación, cree un nodo de destino en el editor visual de AWS Glue Studio.

En este paso, se proporciona el método `write_dynamic_frame.from_options` los parámetros `connection_type`, `connection_options`, `format` y `format_options` para cargar datos en un bucket de destino en Amazon S3.

En el paso 1, inicializó un objeto `GlueContext`. En AWS Glue, aquí encontrarás métodos que se utilizan para configurar los objetivos, al igual que las fuentes.

En este procedimiento, escribirá el siguiente código utilizando `write_dynamic_frame.from_options`. Este código es una parte del script de ejemplo generado.

```
S3bucket_node3 = glueContext.write_dynamic_frame.from_options(
    frame=ApplyMapping_node2,
    connection_type="s3",
    format="glueparquet",
    connection_options={"path": "s3://DOC-EXAMPLE-BUCKET", "partitionKeys": []},
    format_options={"compression": "gzip"},
    transformation_ctx="S3bucket_node3",
    )
```

Para cargar datos en un destino

1. Examine la documentación para buscar un método con el que cargar datos en un bucket de Amazon S3 de destino. Estos métodos están documentados en the section called “GlueContext”. Elija el método the section called “write_dynamic_frame_from_options”. Llame a este método en `glueContext`.

   > **Note**
   >
   > Métodos habituales para cargar datos
   > `write_dynamic_frame.from_options` es el método más utilizado para cargar datos. Es compatible con todos los objetivos disponibles en AWS Glue.
   > Si escribes en un objetivo JDBC definido en una conexión AWS Glue, usa el the section called “write_dynamic_frame_from_jdbc_conf” método. AWS Las conexiones Glue almacenan información sobre cómo conectarse a una fuente de datos. Esto elimina la necesidad de proporcionar esa información en `connection_options`. Sin embargo, aún es necesario utilizar `connection_options` para proporcionar `dbtable`.
   > `write_dynamic_frame.from_catalog` no es un método habitual para cargar datos. Este método actualiza el catálogo de datos de AWS Glue sin actualizar el conjunto

de datos subyacente y se utiliza en combinación con otros procesos que cambian el conjunto de datos subyacente. Para obtener más información, consulte the section called "Cómo actualizar el esquema y añadir nuevas particiones".

2. Examine la documentación de the section called "write_dynamic_frame_from_options". Este método requiere `frame`, `connection_type`, `format`, `connection_options` y `format_options`. Llame a este método en `glueContext`.

   a. Consulte la documentación suplementaria sobre `format_options` y `format` para identificar los parámetros que necesita. Para ver una explicación sobre los formatos de datos, consulte the section called "Opciones de formato de datos".

   b. Consulte la documentación suplementaria sobre `connection_type` y `connection_options` para identificar los parámetros que necesita. Para ver una explicación sobre las conexiones, consulte the section called "Parámetros de conexión".

   c. Proporcione los parámetros necesarios a `write_dynamic_frame.from_options`. Este método tiene una configuración similar a la de `create_dynamic_frame.from_options`.

3. Opcional proporcione `transformation_ctx` a `write_dynamic_frame.from_options` para que se admitan los marcadores de trabajo. Puede obtener información sobre los marcadores de trabajo en la siguiente sección, the section called "Opcional: Habilitar marcadores de trabajo".

Paso 6. Confirmar el objeto **`Job`**

En el paso 1, inicializó un objeto `Job`. En necesario concluir manualmente su ciclo de vida al final del script. Algunas características opcionales necesitan esto para funcionar correctamente. Este trabajo se realiza entre bastidores en AWS Glue Studio.

En este paso, llame al método `commit` del objeto `Job`.

En este procedimiento, escribirá el siguiente código. Este código es una parte del script de ejemplo generado.

```
job.commit()
```

Para confirmar el objeto **`Job`**

1. Si aún no lo ha hecho, realice los pasos opcionales indicados en las secciones anteriores para incluir `transformation_ctx`.

2. Llamar a `commit`.

Opcional: Habilitar marcadores de trabajo

En todos los pasos anteriores, se le ha indicado que configure los parámetros `transformation_ctx`. Esto tiene relación con una característica denominada marcadores de trabajo.

Con los marcadores de trabajos, puede ahorrar tiempo y dinero con los trabajos que se ejecutan de forma recurrente, frente a conjuntos de datos en los que el trabajo anterior puede ser fácilmente rastreado. Los marcadores de trabajo rastrean el progreso de una transformación de AWS Glue en un conjunto de datos de ejecuciones anteriores. Al rastrear dónde terminaron las tiradas anteriores, AWS Glue puede limitar su trabajo a las filas que no haya procesado antes. Para más información acerca de los marcadores de trabajo, consulte the section called “Seguimiento de los datos procesados mediante marcadores de trabajo”.

Para habilitar los marcadores de trabajo, primero hay que agregar las instrucciones `transformation_ctx` en las funciones proporcionadas, tal como se describe en los ejemplos anteriores. El estado de los marcadores de trabajo se mantiene a través de las ejecuciones. Los parámetros `transformation_ctx` son claves que se usan para acceder a ese estado. Por sí solas, estas instrucciones no servirán de nada. También debe activar la característica en la configuración del trabajo.

En este procedimiento, habilita los marcadores de trabajo mediante AWS Management Console.

Para habilitar los marcadores de trabajo

1. Vaya a la sección Job details (Detalles del trabajo) del trabajo correspondiente.
2. Establezca Job bookmark (Marcador de trabajo) como Enable (Habilitar).

Paso 7. Ejecutar el código en forma de trabajo

En este paso, ejecutará el trabajo para comprobar que ha completado correctamente este tutorial. Esto se hace con el clic de un botón, como en el editor visual de AWS Glue Studio.

Para ejecutar el código en forma de trabajo

1. Elija Untitled job (Trabajo sin título) en la barra de título para editar y establecer el nombre del trabajo.

2. Vaya a la pestaña Job details (Detalles del trabajo). Asigne un Rol de IAM al trabajo. Puedes usar el creado por la AWS CloudFormation plantilla en los requisitos previos del tutorial de AWS Glue Studio. Si ha completado ese tutorial, debería estar disponible como `AWS Glue StudioRole`.
3. Elija Save (Guardar) para guardar el script.
4. Elija Run (Ejecutar) para ejecutar el trabajo.
5. Vaya a la pestaña Ejecuciones para comprobar que el trabajo se ha completado.
6. Vaya a *`DOC-EXAMPLE-BUCKET`*, el destino de `write_dynamic_frame.from_options`. Compruebe que el resultado coincide con sus expectativas.

Para obtener más información sobre la configuración y administración de trabajos, consulte the section called “Suministro de sus propios scripts personalizados”.

Más información

Las bibliotecas y los métodos de Apache Spark están disponibles en los scripts de AWS Glue. Puede consultar la documentación de Spark para saber qué puede hacer con esas bibliotecas incluidas. Para obtener más información, consulte la sección de ejemplos del repositorio de origen de Spark.

AWS Glue 2.0+ incluye varias bibliotecas comunes de Python de forma predeterminada. También hay mecanismos para cargar tus propias dependencias en un trabajo de AWS Glue en un entorno de Scala o Python. Para más información sobre las dependencias de Python, consulte the section called “Bibliotecas Python”.

Para ver más ejemplos de cómo utilizar las funciones de AWS Glue en Python, consultethe section called “Muestras de Python”. Los trabajos de Scala y Python disponen de las mismas características, de modo que los ejemplos para Python pueden servir de inspiración a la hora de realizar un trabajo similar en Scala.

Programe scripts AWS Glue ETL en PySpark

Puede encontrar ejemplos de código y utilidades de Python AWS Glue en el repositorio de AWS Glue muestras del sitio GitHub web.

Uso de Python con AWS Glue

AWS Glue admite una extensión del dialecto de PySpark Python para tareas de extracción, transformación y carga (ETL) de secuencias de comandos. En esta sección se describe cómo utilizar Python en scripts de ETL con la API de AWS Glue.

- Configuración para usar Python con AWS Glue
- Llamada a las API de AWS Glue en Python
- Uso de bibliotecas de Python con AWS Glue
- Ejemplos de código Python en AWS Glue

AWS Glue PySpark extensiones

AWS Glueha creado las siguientes extensiones para el dialecto de PySpark Python.

- Acceso a los parámetros mediante `getResolvedOptions`
- Tipos de extensión PySpark
- Clase DynamicFrame
- Clase DynamicFrameCollection
- Clase DynamicFrameWriter
- DynamicFrameReader clase
- GlueContext clase

AWS Glue PySpark transforma

AWS Glueha creado las siguientes clases de transformación para utilizarlas en las operaciones de PySpark ETL.

- Clase de base GlueTransform
- Clase ApplyMapping
- Clase DropFields
- Clase DropNullFields
- Clase ErrorsAsDynamicFrame
- Clase FillMissingValues

- Clase de filtro
- Clase FindIncrementalMatches
- Clase FindMatches
- Clase FlatMap
- Clase Join
- Clase Map
- Clase MapToCollection
- mergeDynamicFrame
- Clase Relationalize
- Clase RenameField
- Clase ResolveChoice
- Clase SelectFields
- Clase SelectFromCollection
- Clase Spigot
- Clase SplitFields
- Clase SplitRows
- Clase Unbox
- Clase UnnestFrame

Configuración para usar Python con AWS Glue

Utilice Python para desarrollar sus scripts de ETL para trabajos de Spark. Las versiones de Python admitidas para los trabajos de ETL dependen de la versión de AWS Glue del trabajo. Para obtener más información acerca de las versiones de AWS Glue, consulte la Glue version job property.

Para configurar su sistema para usar Python con AWS Glue

Siga estos pasos para instalar Python y poder invocar las API de AWS Glue.

1. Si aún no tiene Python instalado, descárguelo e instálelo desde la página de descargas de Python.org.
2. Instale la AWS Command Line Interface (AWS CLI) tal como se detalla en la documentación de la AWS CLI.

La AWS CLI no es directamente necesaria para usar Python. Sin embargo, su instalación y configuración es una forma cómoda de configurar AWS con sus credenciales de cuenta y comprobar que funcionan.

3. Instale el AWS SDK Python (Boto 3), como se documenta en Boto3 Quickstart.

   Las API de recurso de Boto 3 aún no están disponibles para AWS Glue. Actualmente, solo se pueden usar las API de cliente de Boto 3.

   Para obtener más información sobre Boto 3, consulte Introducción al SDK for Python (Boto3) de AWS.

Puede encontrar ejemplos de código de Python y utilidades para AWS Glue en el repositorio de ejemplos de AWS Glue en el sitio web de GitHub.

Llamada a las API de AWS Glue en Python

Tenga en cuenta que las API de recurso de Boto 3 aún no están disponibles para AWS Glue. Actualmente, solo se pueden usar las API de cliente de Boto 3.

Nombres de API de AWS Glue en Python

Los nombres de API de AWS Glue en Java y otros lenguajes de programación suelen ser CamelCased. Sin embargo, al recibir la llamada de Python, estos nombres genéricos se cambian a minúsculas, con las partes del nombre separadas por caracteres de guion bajo para que sean "más Python". En la documentación de referencia AWS Glue API, estos nombres Python aparecen enumerados entre paréntesis después de los nombres CamelCased genéricos.

Sin embargo, aunque los propios nombres de API de AWS Glue cambian a minúsculas, los nombres de sus parámetros permanecen en mayúsculas. Es importante recordar esto, ya que los parámetros deben transferirse por nombre al llamar a las API de AWS Glue, tal como se describe en la sección siguiente.

Suministro y acceso a los parámetros de Python en AWS Glue

En las llamadas de Python a las API de AWS Glue, es mejor transferir los parámetros explícitamente por nombre. Por ejemplo:

```
job = glue.create_job(Name='sample', Role='Glue_DefaultRole',
                      Command={'Name': 'glueetl',
```

```
                                'ScriptLocation': 's3://my_script_bucket/scripts/
my_etl_script.py'})
```

Resulta útil para comprender que Python crea un diccionario de las tuplas nombre/valor que especifica como argumentos para un script de ETL en Estructura de trabajo o JobRun estructura. Boto 3 los transfiere a continuación a AWS Glue en formato JSON a través de una llamada a la API REST. Esto significa que no puede depender del orden de los argumentos al obtener acceso a ellos en su script.

Por ejemplo, supongamos que inicia `JobRun` en una función del controlador de Lambda de Python y desea especificar varios parámetros. Su código podría tener un aspecto similar al siguiente:

```
from datetime import datetime, timedelta

client = boto3.client('glue')

def lambda_handler(event, context):
  last_hour_date_time = datetime.now() - timedelta(hours = 1)
  day_partition_value = last_hour_date_time.strftime("%Y-%m-%d")
  hour_partition_value = last_hour_date_time.strftime("%-H")

  response = client.start_job_run(
               JobName = 'my_test_Job',
               Arguments = {
                 '--day_partition_key':   'partition_0',
                 '--hour_partition_key':  'partition_1',
                 '--day_partition_value':  day_partition_value,
                 '--hour_partition_value': hour_partition_value } )
```

Para obtener acceso a estos parámetros de forma fiable en su script ETL, especifíquelos por nombre mediante la función `getResolvedOptions` de AWS Glue y, a continuación, obtenga acceso a ellos desde el diccionario resultante:

```
import sys
from awsglue.utils import getResolvedOptions

args = getResolvedOptions(sys.argv,
                          ['JOB_NAME',
                           'day_partition_key',
                           'hour_partition_key',
                           'day_partition_value',
```

```
                            'hour_partition_value'])
print "The day partition key is: ", args['day_partition_key']
print "and the day partition value is: ", args['day_partition_value']
```

Si desea transferir un argumento que es una cadena JSON anidada, para preservar el valor del parámetro a medida que se transfiere a su trabajo de ETL de AWS Glue, debe codificar la cadena de parámetros antes de iniciar la ejecución del trabajo y, a continuación, decodificar la cadena de parámetros antes de hacer referencia a su script de trabajo. Por ejemplo, tenga en cuenta las siguientes cadenas de argumentos:

```
glue_client.start_job_run(JobName = "gluejobname", Arguments={
"--my_curly_braces_string": '{"a": {"b": {"c": [{"d": {"e": 42}}]}}}'
})
```

Para transferir este parámetro correctamente, debe codificar el argumento como una cadena codificada en Base64.

```
import base64
...
sample_string='{"a": {"b": {"c": [{"d": {"e": 42}}]}}}'
sample_string_bytes = sample_string.encode("ascii")

base64_bytes = base64.b64encode(sample_string_bytes)
base64_string = base64_bytes.decode("ascii")
...
glue_client.start_job_run(JobName = "gluejobname", Arguments={
"--my_curly_braces_string": base64_bytes})
...
sample_string_bytes = base64.b64decode(base64_bytes)
sample_string = sample_string_bytes.decode("ascii")
print(f"Decoded string: {sample_string}")
...
```

Ejemplo: crear y ejecutar un trabajo

En el siguiente ejemplo se muestra cómo llamar a las API de AWS Glue con Python, para crear y ejecutar un flujo de trabajo de ETL.

Para crear y ejecutar un flujo de trabajo

1. Cree una instancia del cliente de AWS Glue:

```
import boto3
glue = boto3.client(service_name='glue', region_name='us-east-1',
              endpoint_url='https://glue.us-east-1.amazonaws.com')
```

2. Crear un flujo de trabajo. Debe utilizar `glueetl` como nombre del comando de ETL, tal como se muestra en el siguiente código:

```
myJob = glue.create_job(Name='sample', Role='Glue_DefaultRole',
                        Command={'Name': 'glueetl',
                                 'ScriptLocation': 's3://my_script_bucket/
scripts/my_etl_script.py'})
```

3. Inicie una nueva ejecución del flujo de trabajo que creó en el paso anterior:

```
myNewJobRun = glue.start_job_run(JobName=myJob['Name'])
```

4. Obtenga el estado del flujo de trabajo:

```
status = glue.get_job_run(JobName=myJob['Name'], RunId=myNewJobRun['JobRunId'])
```

5. Imprima el estado actual de la ejecución del flujo de trabajo:

```
print(status['JobRun']['JobRunState'])
```

Uso de bibliotecas de Python con AWS Glue

AWS Glue le permite instalar módulos y bibliotecas adicionales de Python para su uso con ETL de AWS Glue.

Temas

- Instalación de módulos de Python adicionales con pip en AWS Glue 2.0+
- Incluir archivos de Python con funciones PySpark nativas
- Scripts de programación que utilizan transformaciones visuales
- Módulos de Python ya proporcionados en AWS Glue
- Compresión de bibliotecas para inclusión
- Carga de bibliotecas de Python en las libretas de AWS Glue Studio
- Carga de bibliotecas Python en un punto de conexión de desarrollo

- Usar bibliotecas de Python en un trabajo o JobRun

Instalación de módulos de Python adicionales con pip en AWS Glue 2.0+

AWS Glue utiliza el instalador de paquetes de Python (pip3) para instalar los módulos adicionales que serán utilizados por ETL de AWS Glue. Puede utilizar el parámetro `--additional-python-modules` con una lista de módulos de Python separados por comas para agregar un nuevo módulo o cambiar la versión de un módulo existente. Puede instalar distribuciones personalizadas de una biblioteca cargando la distribución en Amazon S3, y luego incluir la ruta al objeto de Amazon S3 en la lista de módulos.

Puede pasar opciones adicionales a pip3 con el parámetro `--python-modules-installer-option`. Por ejemplo, puede pasar `"--upgrade"` para actualizar los paquetes especificados por `"--additional-python-modules"`. Para ver más ejemplos, consulta Cómo crear módulos de Python a partir de una rueda para cargas de trabajo ETL de Spark mediante AWS Glue 2.0.

Si tus dependencias de Python dependen transitoriamente del código compilado nativo, es posible que te enfrentes a la siguiente limitación: AWS Glue no admite la compilación de código nativo en el entorno de trabajo. Sin embargo, los trabajos de AWS Glue se ejecutan en un entorno Amazon Linux 2. Es posible que pueda proporcionar las dependencias nativas en forma compilada a través de un archivo distribuible Wheel.

Por ejemplo, para actualizar o agregar un nuevo módulo `scikit-learn` utilice la siguiente clave/valor: `"--additional-python-modules", "scikit-learn==0.21.3"`.

Además, dentro de la opción `--additional-python-modules`, puede especificar una ruta de Amazon S3 a un módulo de wheel de Python. Por ejemplo:

```
--additional-python-modules s3://aws-glue-native-spark/tests/j4.2/ephem-3.7.7.1-cp37-cp37m-linux_x86_64.whl,s3://aws-glue-native-spark/tests/j4.2/fbprophet-0.6-py3-none-any.whl,scikit-learn==0.21.3
```

Se especifica `--additional-python-modules` en el campo Parámetros del trabajo de la AWS Glue consola o modificando los argumentos del trabajo en el AWS SDK. Para obtener más información sobre la configuración de parámetros de trabajos, consulte the section called “Parámetros del flujo de trabajo”.

Incluir archivos de Python con funciones PySpark nativas

AWS Glue PySpark se utiliza para incluir archivos de Python en los trabajos de ETL de AWS Glue. Podrá utilizar `--additional-python-modules` para administrar las dependencias cuando esté disponible. Puede utilizar el parámetro de trabajo `--extra-py-files` para incluir archivos de Python. Las dependencias deben estar alojadas en Amazon S3 y el valor del argumento debe ser una lista delimitada por comas de rutas de Amazon S3 sin espacios. Esta funcionalidad se comporta igual que la administración de dependencias de Python que se utilizaría con Spark. Para obtener más información sobre la gestión de dependencias de Python en Spark, consulta la página Uso de funciones PySpark nativas en la documentación de Apache Spark. `--extra-py-files`es útil en los casos en los que tu código adicional no está empaquetado o cuando estás migrando un programa de Spark con una cadena de herramientas existente para gestionar las dependencias. Para que las herramientas para dependencias sean mantenibles, deberá agrupar las dependencias antes del envío.

Scripts de programación que utilizan transformaciones visuales

Al crear un trabajo de AWS Glue mediante la interfaz visual de AWS Glue Studio, puede transformar sus datos con nodos de transformación de datos gestionados y transformaciones visuales personalizadas. Par más información sobre los nodos administrados de transformación de datos, consulte the section called “Edición de nodos de transformación de datos administrados por AWS Glue”. Para obtener más información sobre las transformaciones visuales personalizadas, consulte the section called “ Transformaciones visuales personalizadas ”. Los scripts que utilizan transformaciones visuales solo se pueden generar cuando el Lenguaje del trabajo está establecido para que utilice Python.

Al generar un trabajo de AWS Glue mediante transformaciones visuales, AWS Glue Studio incluirá estas transformaciones en el entorno de ejecución mediante el `--extra-py-files` parámetro de la configuración del trabajo. Para obtener más información acerca de la configuración de parámetros de trabajos, consulte the section called “Parámetros del flujo de trabajo”. Cuando se realizan cambios en un script generado o un entorno de tiempo de ejecución, necesitará conservar la configuración del trabajo para que script se ejecute de manera exitosa.

Módulos de Python ya proporcionados en AWS Glue

Para cambiar la versión de estos módulos proporcionados, indique las nuevas versiones con el parámetro de trabajo `--additional-python-modules`.

AWS Glue version 2.0

La versión 2.0 AWS Glue incluye los siguientes módulos de Python listos para su uso:

- avro-python3==1.10.0
- awscli==1.27.60
- boto3==1.12.4
- botocore==1.15.4
- certifi==2019.11.28
- chardet==3.0.4
- click==8.1.3
- colorama==0.4.4
- cycler==0.10.0
- Cython==0.29.15
- docutils==0.15.2
- enum34==1.1.9
- fsspec==0.6.2
- idna==2.9
- importlib-metadata==6.0.0
- jmespath==0.9.4
- Joblib==0.14.1
- kiwisolver==1.1.0
- matplotlib==3.1.3
- mpmath==1.1.0
- nltk==3.5
- numpy==1.18.1
- pandas==1.0.1
- patsy==0.5.1
- pmdarima==1.5.3
- ptvsd==4.3.2

- pyarrow==0.16.0
- pyasn1==0.4.8
- pydevd==1.9.0
- pyhocon==0.3.54
- PyMySQL==0.9.3
- pyparsing==2.4.6
- python-dateutil==2.8.1
- pytz==2019.3
- PyYAML==5.3.1
- regex==2022.10.31
- requests==2.23.0
- rsa==4.7.2
- s3fs==0.4.0
- s3transfer==0.3.3
- scikit-learn==0.22.1
- scipy==1.4.1
- setuptools==45.2.0
- six==1.14.0
- Spark==1.0
- statsmodels==0.11.1
- subprocess32==3.5.4
- sympy==1.5.1
- tbats==1.0.9
- tqdm==4.64.1
- typing-extensions==4.4.0
- urllib3==1.25.8
- wheel==0.35.1
- zipp==3.12.0

AWS Glue versión 3.0

La versión 3.0 AWS Glue incluye los siguientes módulos de Python listos para su uso:

- aiobotocore==1.4.2
- aiohttp==3.8.3
- aioitertools==0.11.0
- aiosignal==1.3.1
- async-timeout==4.0.2
- asynctest==0.13.0
- attrs==22.2.0
- avro-python3==1.10.2
- boto3==1.18.50
- botocore==1.21.50
- certifi==2021.5.30
- chardet==3.0.4
- charset-normalizer==2.1.1
- click==8.1.3
- cycler==0.10.0
- Cython==0.29.4
- docutils==0.17.1
- enum34==1.1.10
- frozenlist==1.3.3
- fsspec==2021.8.1
- idna==2.10
- importlib-metadata==6.0.0
- jmespath==0.10.0
- joblib==1.0.1
- kiwisolver==1.3.2
- matplotlib==3.4.3

- mpmath==1.2.1
- multidict==6.0.4
- nltk==3.6.3
- numpy==1.19.5
- packaging==23.0
- pandas==1.3.2
- patsy==0.5.1
- Pillow==9.4.0
- pip==23.0
- pmdarima==1.8.2
- ptvsd==4.3.2
- pyarrow==5.0.0
- pydevd==2.5.0
- pyhocon==0.3.58
- PyMySQL==1.0.2
- pyparsing==2.4.7
- python-dateutil==2.8.2
- pytz==2021.1
- PyYAML==5.4.1
- regex==2022.10.31
- requests==2.23.0
- s3fs==2021.8.1
- s3transfer==0.5.0
- scikit-learn==0.24.2
- scipy==1.7.1
- six==1.16.0
- Spark==1.0
- statsmodels==0.12.2
- subprocess32==3.5.4

- sympy==1.8
- tbats==1.1.0
- threadpoolctl==3.1.0
- tqdm==4.64.1
- typing_extensions==4.4.0
- urllib3==1.25.11
- wheel==0.37.0
- wrapt==1.14.1
- yarl==1.8.2
- zipp==3.12.0

AWS Glue versión 4.0

La versión 4.0 AWS Glue incluye los siguientes módulos de Python listos para su uso:

- aiobotocore==2.4.1
- aiohttp==3.8.3
- aioitertools==0.11.0
- aiosignal==1.3.1
- async-timeout==4.0.2
- asynctest==0.13.0
- attrs==22.2.0
- avro-python3==1.10.2
- boto3==1.24.70
- botocore==1.27.59
- certifi==2021.5.30
- chardet==3.0.4
- charset-normalizer==2.1.1
- click==8.1.3
- cycler==0.10.0
- Cython==0.29.32

- docutils==0.17.1
- enum34==1.1.10
- frozenlist==1.3.3
- fsspec==2021.8.1
- idna==2.10
- importlib-metadata==5.0.0
- jmespath==0.10.0
- joblib==1.0.1
- kaleido==0.2.1
- kiwisolver==1.4.4
- matplotlib==3.4.3
- mpmath==1.2.1
- multidict==6.0.4
- nltk==3.7
- numpy==1.23.5
- packaging==23.0
- pandas==1.5.1
- patsy==0.5.1
- Pillow==9.4.0
- pip==23.0.1
- plotly==5.16.0
- pmdarima==2.0.1
- ptvsd==4.3.2
- pyarrow==10.0.0
- pydevd==2.5.0
- pyhocon==0.3.58
- PyMySQL==1.0.2
- pyparsing==2.4.7
- python-dateutil==2.8.2

- pytz==2021.1
- PyYAML==6.0.1
- regex==2022.10.31
- requests==2.23.0
- s3fs==2022.11.0
- s3transfer==0.6.0
- scikit-learn==1.1.3
- scipy==1.9.3
- setuptools==49.1.3
- six==1.16.0
- statsmodels==0.13.5
- subprocess32==3.5.4
- sympy==1.8
- tbats==1.1.0
- threadpoolctl==3.1.0
- tqdm==4.64.1
- typing_extensions==4.4.0
- urllib3==1.25.11
- wheel==0.37.0
- wrapt==1.14.1
- yarl==1.8.2
- zipp==3.10.0

Compresión de bibliotecas para inclusión

Salvo que una biblioteca se encuentre en un único archivo `.py`, deberá empaquetarse en un archivo `.zip`. El directorio del paquete debe encontrarse en la raíz del archivo y debe contener un archivo `__init__.py` para el paquete. Posteriormente, Python podrá importar el paquete de la forma habitual.

Si la biblioteca se compone solamente de un único módulo de Python en un archivo `.py`, no será necesario comprimirlo en un archivo `.zip`.

Carga de bibliotecas de Python en las libretas de AWS Glue Studio

Para especificar las bibliotecas de Python en los cuadernos de AWS Glue Studio, consulte Instalar módulos de Python adicionales.

Carga de bibliotecas Python en un punto de conexión de desarrollo

Si utiliza diferentes conjuntos de bibliotecas para distintos scripts de ETL, puede configurar puntos de enlace de desarrollo independientes para cada conjunto o bien, puede sobrescribir los archivos `.zip` de la biblioteca que carga el punto de enlace de desarrollo cada vez que usted cambia de script.

Puede utilizar la consola con el fin de especificar uno o varios archivos .zip para un punto de enlace de desarrollo cuando lo cree. Después de asignar un nombre y un rol de IAM, seleccione Script Libraries and job parameters (Bibliotecas de script y parámetros de trabajo) (opcional) y escriba la ruta de Amazon S3 completa para los archivos `.zip` de la biblioteca en el cuadro Python library path (Ruta de la biblioteca Python). Por ejemplo:

```
s3://bucket/prefix/site-packages.zip
```

También puede especificar varias rutas completas a los archivos; para ello, sepárelas con comas pero sin espacios, como sigue:

```
s3://bucket/prefix/lib_A.zip,s3://bucket_B/prefix/lib_X.zip
```

Si actualiza estos archivos `.zip` más adelante, puede utilizar la consola para volver a importarlos en el punto de enlace de desarrollo. Vaya al punto de enlace del desarrollador que corresponda, marque la casilla junto al mismo y seleccione Update ETL libraries (Actualizar bibliotecas de ETL) desde el menú Action (Acción).

De forma similar, puede especificar archivos de biblioteca mediante las API de AWS Glue. Cuando crea un punto de enlace de desarrollo al invocar Acción CreateDevEndpoint (Python: create_dev_endpoint), puede especificar una o varias rutas completas para las bibliotecas en el parámetro `ExtraPythonLibsS3Path`, en una llamada con un formato similar al siguiente:

```
dep = glue.create_dev_endpoint(
             EndpointName="testDevEndpoint",
             RoleArn="arn:aws:iam::123456789012",
             SecurityGroupIds="sg-7f5ad1ff",
             SubnetId="subnet-c12fdba4",
             PublicKey="ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQCtp04H/y...",
```

```
            NumberOfNodes=3,
            ExtraPythonLibsS3Path="s3://bucket/prefix/lib_A.zip,s3://bucket_B/prefix/
lib_X.zip")
```

Cuando actualice un punto de enlace de desarrollo, también puede actualizar las bibliotecas que carga con un objeto DevEndpointCustomLibraries y si establece el parámetro `UpdateEtlLibraries` en `True` a la hora de invocar UpdateDevEndpoint (update_dev_endpoint).

Usar bibliotecas de Python en un trabajo o JobRun

Cuando crea un trabajo nuevo en la consola, puede especificar uno o más archivos .zip de la biblioteca si selecciona Script Libraries and job parameters (Bibliotecas de script y parámetros de trabajo) (opcional) y escribe las rutas completas de la biblioteca de Amazon S3, igual que cuando crea un punto de enlace de desarrollo:

```
s3://bucket/prefix/lib_A.zip,s3://bucket_B/prefix/lib_X.zip
```

Si invoca CreateJob (create_job), puede especificar una o varias rutas completas para las bibliotecas predeterminadas mediante el parámetro predeterminado `--extra-py-files` , como se indica a continuación:

```
job = glue.create_job(Name='sampleJob',
                      Role='Glue_DefaultRole',
                      Command={'Name': 'glueetl',
                               'ScriptLocation': 's3://my_script_bucket/scripts/
my_etl_script.py'},
                      DefaultArguments={'--extra-py-files': 's3://bucket/prefix/
lib_A.zip,s3://bucket_B/prefix/lib_X.zip'})
```

Luego, al iniciar una JobRun, puede anular la configuración de biblioteca predeterminada por otra diferente:

```
runId = glue.start_job_run(JobName='sampleJob',
                           Arguments={'--extra-py-files': 's3://bucket/prefix/
lib_B.zip'})
```

Ejemplos de código Python en AWS Glue

- Código de ejemplo: unión de los datos y establecimiento de relaciones entre ellos
- Ejemplo de código: Preparación de datos con ResolveChoice, Lambda y ApplyMapping

Código de ejemplo: unión de los datos y establecimiento de relaciones entre ellos

En este ejemplo se utiliza un conjunto de datos que se ha descargado desde http://everypolitician.org/ al bucket `sample-dataset` de Amazon Simple Storage Service (Amazon S3): `s3://awsglue-datasets/examples/us-legislators/all`. El conjunto de datos contiene información en formato JSON sobre legisladores de los Estados Unidos y los escaños de los que han sido titulares en la Cámara de Representantes y el Senado. Para realizar este tutorial, este conjunto de datos se ha modificado ligeramente y está disponible en un bucket público de Amazon S3.

Puede encontrar el código fuente de este ejemplo en el archivo `join_and_relationalize.py` del repositorio de ejemplos de AWS Glue en el sitio web de GitHub.

Con estos datos, el presente tutorial le enseña a realizar las siguientes tareas:

- Utilice un rastreador de AWS Glue para clasificar los objetos que están almacenados en un bucket público de Amazon S3 y guardar sus esquemas en el Catálogo de datos de AWS Glue.
- Examinar los metadatos y los esquemas de la tabla que se obtienen a partir del rastreo.
- Escriba un script de extracción, transferencia y carga (ETL) de Python que utilice los metadatos del Catálogo de datos para hacer lo siguiente:
  - Unir los datos de los diferentes archivos de origen juntos en una única tabla de datos (es decir, desnormalizar los datos).
  - Desglosar la tabla unida en diferentes tablas según el tipo de legislador.
  - Escribir los datos resultantes en archivos Apache Parquet independientes para realizar un análisis posteriormente.

La forma preferida de depurar scripts de Python o PySpark mientras se están ejecutando en AWS es usar Notebooks en Glue Studio AWS.

Paso 1: Rastrear los datos del bucket de Amazon S3

1. Luego, inicie sesión en la AWS Management Console y abra la consola de AWS Glue en https://console.aws.amazon.com/glue/.
2. En función de los pasos que se indican en Configuración de rastreadores, cree un nuevo rastreador que pueda rastrear el conjunto de datos `s3://awsglue-datasets/examples/us-legislators/all` en una base de datos denominada `legislators` en el Catálogo de datos de AWS Glue. Los datos de ejemplo ya están en este bucket público de Amazon S3.
3. Ejecute el nuevo rastreador y, a continuación, compruebe la base de datos `legislators`.

El rastreador crea las siguientes tablas de metadatos:

- `persons_json`
- `memberships_json`
- `organizations_json`
- `events_json`
- `areas_json`
- `countries_r_json`

Esta es una colección seminormalizada de tablas que contienen legisladores y sus historias.

Paso 2: Añadir un script reutilizable al cuaderno del punto de conexión de desarrollo

Pegue el script reutilizable siguiente en el cuaderno del punto de enlace de desarrollo para importar las bibliotecas de AWS Glue que necesite y configurar un único `GlueContext`:

```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job

glueContext = GlueContext(SparkContext.getOrCreate())
```

Paso 3: Examinar los esquemas de los datos del Catálogo de datos

A continuación, puede crear un DynamicFrame con facilidad desde el Catálogo de datos de AWS Glue y examinar los esquemas de los datos. Por ejemplo, para ver el esquema de la tabla `persons_json`, añada lo siguiente en su cuaderno:

```
persons = glueContext.create_dynamic_frame.from_catalog(
             database="legislators",
             table_name="persons_json")
print "Count: ", persons.count()
persons.printSchema()
```

Esta es la salida de las llamadas impresas:

```
Count:  1961
root
|-- family_name: string
|-- name: string
|-- links: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- url: string
|-- gender: string
|-- image: string
|-- identifiers: array
|    |-- element: struct
|    |    |-- scheme: string
|    |    |-- identifier: string
|-- other_names: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- name: string
|    |    |-- lang: string
|-- sort_name: string
|-- images: array
|    |-- element: struct
|    |    |-- url: string
|-- given_name: string
|-- birth_date: string
|-- id: string
|-- contact_details: array
|    |-- element: struct
|    |    |-- type: string
|    |    |-- value: string
|-- death_date: string
```

Cada persona de la tabla es miembro de algún órgano del congreso de los Estados Unidos.

Para ver el esquema de la tabla `memberships_json`, escriba lo siguiente:

```
memberships = glueContext.create_dynamic_frame.from_catalog(
                database="legislators",
                table_name="memberships_json")
```

```
print "Count: ", memberships.count()
memberships.printSchema()
```

La salida es la siguiente:

```
Count:  10439
root
|-- area_id: string
|-- on_behalf_of_id: string
|-- organization_id: string
|-- role: string
|-- person_id: string
|-- legislative_period_id: string
|-- start_date: string
|-- end_date: string
```

Las `organizations` son partes y las dos cámaras del Congreso, el Senado y la Cámara de Representantes. Para ver el esquema de la tabla `organizations_json`, escriba lo siguiente:

```
orgs = glueContext.create_dynamic_frame.from_catalog(
           database="legislators",
           table_name="organizations_json")
print "Count: ", orgs.count()
orgs.printSchema()
```

La salida es la siguiente:

```
Count:  13
root
|-- classification: string
|-- links: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- url: string
|-- image: string
|-- identifiers: array
|    |-- element: struct
|    |    |-- scheme: string
|    |    |-- identifier: string
```

```
|-- other_names: array
|    |-- element: struct
|    |    |-- lang: string
|    |    |-- note: string
|    |    |-- name: string
|-- id: string
|-- name: string
|-- seats: int
|-- type: string
```

Paso 4: Filtrar los datos

A continuación, mantenga solo los campos que desee y cambie el nombre `id` por `org_id`. El conjunto de datos es lo suficientemente pequeño para ver todo el conjunto.

`toDF()` convierte a `DynamicFrame` en un elemento `DataFrame` de Apache Spark, por lo que puede aplicar las transformaciones que ya existen en Apache Spark SQL:

```
orgs = orgs.drop_fields(['other_names',
                         'identifiers']).rename_field(
                             'id', 'org_id').rename_field(
                                'name', 'org_name')
orgs.toDF().show()
```

El ejemplo siguiente muestra la salida:

```
+--------------+--------------------+--------------------+--------------------+-----
+-----------+--------------------+
|classification|              org_id|            org_name|               links|seats|
     type|               image|
+--------------+--------------------+--------------------+--------------------+-----
+-----------+--------------------+
|         party|            party/al|                  AL|                null| null|
     null|                null|
|         party|      party/democrat|            Democrat|[[website,http://...| null|
     null|https://upload.wi...|
|         party|party/democrat-li...|    Democrat-Liberal|[[website,http://...| null|
     null|                null|
|   legislature|d56acebe-8fdc-47b...|House of Represen...|                null|  435|
lower house|                null|
```

```
|         party|   party/independent|         Independent|                null| null|
     null|                null|
|         party|party/new_progres...|     New Progressive|[[website,http://...| null|
     null|https://upload.wi...|
|         party|party/popular_dem...|    Popular Democrat|[[website,http://...| null|
     null|                null|
|         party|    party/republican|          Republican|[[website,http://...| null|
     null|https://upload.wi...|
|         party|party/republican-...|Republican-Conser...|[[website,http://...| null|
     null|                null|
|         party|      party/democrat|            Democrat|[[website,http://...| null|
     null|https://upload.wi...|
|         party|   party/independent|         Independent|                null| null|
     null|                null|
|         party|    party/republican|          Republican|[[website,http://...| null|
     null|https://upload.wi...|
|   legislature|8fa6c3d2-71dc-478...|              Senate|                null|  100|
upper house|                null|
+--------------+--------------------+--------------------+--------------------+-----
+-----------+--------------------+
```

Escriba lo siguiente para ver las `organizations` que aparecen en `memberships`:

```
memberships.select_fields(['organization_id']).toDF().distinct().show()
```

El ejemplo siguiente muestra la salida:

```
+--------------------+
|     organization_id|
+--------------------+
|d56acebe-8fdc-47b...|
|8fa6c3d2-71dc-478...|
+--------------------+
```

Paso 5: Reunirlo todo

Ahora utilice AWS Glue para unir estas tablas relacionales y crear una tabla del historial completo de `memberships` del legislador y sus `organizations` correspondientes.

1. En primer lugar, una `persons` y `memberships` en `id` y `person_id`.

2. A continuación, una el resultado con `orgs` en `org_id` y `organization_id`.
3. A continuación, anule los campos redundantes `person_id` y `org_id`.

Puede realizar todas estas operaciones en una línea de código (extendida):

```
l_history = Join.apply(orgs,
                       Join.apply(persons, memberships, 'id', 'person_id'),
                       'org_id', 'organization_id').drop_fields(['person_id',
 'org_id'])
print "Count: ", l_history.count()
l_history.printSchema()
```

La salida es la siguiente:

```
Count:  10439
root
|-- role: string
|-- seats: int
|-- org_name: string
|-- links: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- url: string
|-- type: string
|-- sort_name: string
|-- area_id: string
|-- images: array
|    |-- element: struct
|    |    |-- url: string
|-- on_behalf_of_id: string
|-- other_names: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- name: string
|    |    |-- lang: string
|-- contact_details: array
|    |-- element: struct
|    |    |-- type: string
|    |    |-- value: string
|-- name: string
```

```
|-- birth_date: string
|-- organization_id: string
|-- gender: string
|-- classification: string
|-- death_date: string
|-- legislative_period_id: string
|-- identifiers: array
|    |-- element: struct
|    |    |-- scheme: string
|    |    |-- identifier: string
|-- image: string
|-- given_name: string
|-- family_name: string
|-- id: string
|-- start_date: string
|-- end_date: string
```

Ahora tiene la tabla definitiva que puede utilizar para su análisis. Puede escribirla en un formato compacto y eficiente para el análisis, por ejemplo, en Parquet, en el que puede ejecutar SQL en AWS Glue, Amazon Athena o Amazon Redshift Spectrum.

La siguiente llamada escribe la tabla en varios archivos para admitir lecturas paralelas rápidas al realizar el análisis posterior:

```
glueContext.write_dynamic_frame.from_options(frame = l_history,
          connection_type = "s3",
          connection_options = {"path": "s3://glue-sample-target/output-dir/
legislator_history"},
          format = "parquet")
```

Para poner todos los datos de historial en un único archivo, debe convertirlos en una estructura de datos, crear nuevas particiones y escribirlos:

```
s_history = l_history.toDF().repartition(1)
s_history.write.parquet('s3://glue-sample-target/output-dir/legislator_single')
```

O, si desea separarlos por Senado y Cámara:

```
l_history.toDF().write.parquet('s3://glue-sample-target/output-dir/legislator_part',
```

```
                                    partitionBy=['org_name'])
```

Paso 6: Transformar los datos para bases de datos relacionales

AWS Glue facilita la tarea de escribir los datos en bases de datos relacionales como Amazon Redshift, incluso con datos semiestructurados. Ofrece una transformación `relationalize`, que aplana `DynamicFrames` sea cual sea la complejidad de los objetos de la trama.

Utilizando el elemento `l_history` `DynamicFrame` de este ejemplo, pase el nombre de una tabla (`hist_root`) y una ruta de flujo de trabajo temporal a `relationalize`. Esto devuelve un elemento `DynamicFrameCollection`. A continuación, puede enumerar los nombres de `DynamicFrames` en esa colección:

```
dfc = l_history.relationalize("hist_root", "s3://glue-sample-target/temp-dir/")
dfc.keys()
```

A continuación, se muestra la salida de la llamada `keys`:

```
[u'hist_root', u'hist_root_contact_details', u'hist_root_links',
 u'hist_root_other_names', u'hist_root_images', u'hist_root_identifiers']
```

`Relationalize` ha desglosado la tabla de historial en seis tablas nuevas: una tabla raíz que contiene un registro por cada objeto de `DynamicFrame` y tablas auxiliares para las matrices. A menudo, la gestión de matrices en las bases de datos relacionales no tiene un nivel óptimo, en particular cuando dichas matrices se hacen más grandes. Si se separan las matrices en diferentes tablas, las consultas serán mucho más rápidas.

A continuación, céntrese en la separación examinando `contact_details`:

```
l_history.select_fields('contact_details').printSchema()
dfc.select('hist_root_contact_details').toDF().where("id = 10 or id =
 75").orderBy(['id','index']).show()
```

A continuación, se muestra la salida de la llamada `show`:

```
root
```

```
|-- contact_details: array
|    |-- element: struct
|    |    |-- type: string
|    |    |-- value: string
+---+-----+-----------------------+-------------------------+
| id|index|contact_details.val.type|contact_details.val.value|
+---+-----+-----------------------+-------------------------+
| 10|    0|                    fax|                         |
| 10|    1|                       |             202-225-1314|
| 10|    2|                  phone|                         |
| 10|    3|                       |             202-225-3772|
| 10|    4|                twitter|                         |
| 10|    5|                       |          MikeRossUpdates|
| 75|    0|                    fax|                         |
| 75|    1|                       |             202-225-7856|
| 75|    2|                  phone|                         |
| 75|    3|                       |             202-225-2711|
| 75|    4|                twitter|                         |
| 75|    5|                       |                SenCapito|
+---+-----+-----------------------+-------------------------+
```

El campo `contact_details` era una matriz de estructuras en el elemento `DynamicFrame` original. Cada elemento de estas matrices es una fila independiente de la tabla auxiliar, indizada por `index`. Aquí el `id` es una clave externa en la tabla `hist_root` con la clave `contact_details`:

```
dfc.select('hist_root').toDF().where(
    "contact_details = 10 or contact_details = 75").select(
       ['id', 'given_name', 'family_name', 'contact_details']).show()
```

Se genera la salida siguiente:

```
+--------------------+----------+-----------+---------------+
|                  id|given_name|family_name|contact_details|
+--------------------+----------+-----------+---------------+
|f4fc30ee-7b42-432...|      Mike|       Ross|             10|
|e3c60f34-7d1b-4c0...|   Shelley|     Capito|             75|
+--------------------+----------+-----------+---------------+
```

Observe en estos comandos que se utiliza `toDF()` y, a continuación, una expresión `where`, para filtrar y obtener las filas que desea ver.

Por lo tanto, unir la tabla `hist_root` con las tablas auxiliares le permite hacer lo siguiente:

- Cargar datos en bases de datos sin compatibilidad de matrices.
- Consultar cada elemento individual de una matriz con SQL.

Almacene las credenciales de Amazon Redshift y acceda a ellas de forma segura con una conexión de AWS Glue. Para obtener información acerca de cómo crear su propia conexión, consulte Conexión a datos.

Ahora está preparado para escribir los datos en una conexión pasando por los elementos `DynamicFrames` de uno en uno:

```
for df_name in dfc.keys():
  m_df = dfc.select(df_name)
  print "Writing to table: ", df_name
  glueContext.write_dynamic_frame.from_jdbc_conf(frame = m_df, connection settings
 here)
```

La configuración de la conexión variará en función del tipo de base de datos relacional:

- Para obtener instrucciones sobre cómo escribir en Amazon Redshift, consulte the section called “Conexiones Redshift”.
- Para otras bases de datos, consulte Tipos de conexión y opciones para ETL en AWS Glue para Spark.

Conclusión

En general, AWS Glue es muy flexible. Le permite realizar en unas cuantas líneas de código lo que normalmente tardaría varios días en escribirse. Puede encontrar todos los scripts de ETL de origen a destino en el archivo de Python `join_and_relationalize.py`, en los ejemplos de AWS Glue en GitHub.

Ejemplo de código: Preparación de datos con ResolveChoice, Lambda y ApplyMapping

El conjunto de datos que se utiliza en este ejemplo está formado por datos de pago de Medicare Provider que se descargaron de dos sitios de Data.CMS.gov, conjuntos de datos: "Inpatient Prospective Payment System Provider Summary for the Top 100 Diagnosis-Related Groups -

FY2011" e "Inpatient Charge Data FY 2011". Después de descargar los datos, modificamos el conjunto de datos para presentar un par de registros erróneos al final del archivo. Este archivo modificado se encuentra en un bucket público de Amazon S3 en `s3://awsglue-datasets/examples/medicare/Medicare_Hospital_Provider.csv`.

Puede encontrar el código fuente de este ejemplo en el archivo `data_cleaning_and_lambda.py` del repositorio GitHub de ejemplos de AWS Glue.

La forma preferida de depurar scripts de Python o PySpark mientras se están ejecutando en AWS es usar Notebooks en Glue Studio AWS.

Paso 1: Rastrear los datos del bucket de Amazon S3

1. Inicie sesión en la AWS Management Console y abra la consola de AWS Glue en https://console.aws.amazon.com/glue/.
2. En función del proceso descrito en Configuración de rastreadores, cree un rastreador nuevo que pueda rastrear el archivo `s3://awsglue-datasets/examples/medicare/Medicare_Hospital_Provider.csv` y poner los metadatos resultantes en una base de datos denominada `payments` en AWS Glue Data Catalog.
3. Ejecute el nuevo rastreador y, a continuación, compruebe la base de datos `payments`. Deberá ver que el rastreador ha creado en la base de datos una tabla de metadatos denominada `medicare` después de leer el comienzo del archivo para determinar su formato y delimitador.

   El esquema de la tabla `medicare` nueva es el siguiente:

```
Column  name                                Data type
================================================
drg definition                                string
provider id                                   bigint
provider name                                 string
provider street address                       string
provider city                                 string
provider state                                string
provider zip code                             bigint
hospital referral region description          string
total discharges                              bigint
average covered charges                       string
average total payments                        string
average medicare payments                     string
```

Paso 2: Añadir un script reutilizable al cuaderno del punto de conexión de desarrollo

Pegue el script reutilizable siguiente en el cuaderno del punto de enlace de desarrollo para importar las bibliotecas de AWS Glue que necesite y configurar un único `GlueContext`:

```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job

glueContext = GlueContext(SparkContext.getOrCreate())
```

Paso 3: Comparar diferentes análisis de esquemas

A continuación, puede ver si el esquema que un elemento `DataFrame` de Apache Spark reconoció es el mismo que su rastreador AWS Glue ha registrado. Ejecute el código siguiente:

```
medicare = spark.read.format(
   "com.databricks.spark.csv").option(
   "header", "true").option(
   "inferSchema", "true").load(
   's3://awsglue-datasets/examples/medicare/Medicare_Hospital_Provider.csv')
medicare.printSchema()
```

Esta es la salida de la llamada `printSchema`:

```
root
 |-- DRG Definition: string (nullable = true)
 |-- Provider Id: string (nullable = true)
 |-- Provider Name: string (nullable = true)
 |-- Provider Street Address: string (nullable = true)
 |-- Provider City: string (nullable = true)
 |-- Provider State: string (nullable = true)
 |-- Provider Zip Code: integer (nullable = true)
 |-- Hospital Referral Region Description: string (nullable = true)
 |--  Total Discharges : integer (nullable = true)
 |--  Average Covered Charges : string (nullable = true)
 |--  Average Total Payments : string (nullable = true)
```

```
 |-- Average Medicare Payments: string (nullable = true)
```

A continuación, examine el esquema que un `DynamicFrame` de AWS Glue genera:

```
medicare_dynamicframe = glueContext.create_dynamic_frame.from_catalog(
       database = "payments",
       table_name = "medicare")
medicare_dynamicframe.printSchema()
```

La salida de `printSchema` es la siguiente:

```
root
 |-- drg definition: string
 |-- provider id: choice
 |    |-- long
 |    |-- string
 |-- provider name: string
 |-- provider street address: string
 |-- provider city: string
 |-- provider state: string
 |-- provider zip code: long
 |-- hospital referral region description: string
 |-- total discharges: long
 |-- average covered charges: string
 |-- average total payments: string
 |-- average medicare payments: string
```

El elemento `DynamicFrame` genera un esquema donde `provider id` puede ser el tipo `long` o el tipo `string`. El esquema `DataFrame` indica que `Provider Id` es de tipo `string`, y el Data Catalog indica que `provider id` es de tipo `bigint`.

¿Cuál es el correcto? Hay dos registros al final del archivo (de 160 000 registros) con valores `string` en dicha columna. Estos son los registros erróneos que se introdujeron para ilustrar un problema.

Para abordar este tipo de problema, el `DynamicFrame` de AWS Glue introduce el concepto de tipo de choice (elección). En este caso, `DynamicFrame` muestra que los valores `long` y `string` pueden aparecer en dicha columna. El rastreador de AWS Glue no tuvo en cuenta los valores `string`, ya que consideró solo un prefijo de 2 MB de los datos. El elemento `DataFrame` de Apache Spark tuvo

en cuenta el conjunto de datos en su totalidad, pero se vio obligado a asignar el tipo más general a la columna, en este caso `string`. De hecho, Spark a menudo recurre al caso más general cuando hay tipos complejos o variaciones con las que no está familiarizado.

Para consultar la columna `provider id`, primero debe resolver el tipo de elección. Puede utilizar el método de transformación `resolveChoice` de `DynamicFrame` para convertir estos valores `string` en valores `long` con una opción `cast:long`:

```
medicare_res = medicare_dynamicframe.resolveChoice(specs = [('provider
 id','cast:long')])
medicare_res.printSchema()
```

Ahora la salida `printSchema` es:

```
root
 |-- drg definition: string
 |-- provider id: long
 |-- provider name: string
 |-- provider street address: string
 |-- provider city: string
 |-- provider state: string
 |-- provider zip code: long
 |-- hospital referral region description: string
 |-- total discharges: long
 |-- average covered charges: string
 |-- average total payments: string
 |-- average medicare payments: string
```

Cuando el valor era un elemento `string` que no se podía transformar, AWS Glue insertaba un valor `null`.

Otra opción consiste en convertir el tipo de elección en un elemento `struct`, que mantiene los valores de ambos tipos.

A continuación, examine las filas que eran anómalas:

```
medicare_res.toDF().where("'provider id' is NULL").show()
```

Verá lo siguiente:

```
+--------------------+----------+--------------+----------------------+------------
+-------------+----------------+-----------------------------------+---------------
+----------------------+---------------------+------------------------+
|      drg definition|provider id|  provider name|provider street address|provider
 city|provider state|provider zip code|hospital referral region description|total
 discharges|average covered charges|average total payments|average medicare payments|
+--------------------+----------+--------------+----------------------+------------
+-------------+----------------+-----------------------------------+---------------
+----------------------+---------------------+------------------------+
|948 - SIGNS & SYM...|       null|            INC|       1050 DIVISION ST|
 MAUSTON|            WI|            53948|                        WI - Madison|
      12|              $11961.41|              $4619.00|                $3775.33|
|948 - SIGNS & SYM...|       null| INC- ST JOSEPH|     5000 W CHAMBERS ST|
 MILWAUKEE|            WI|            53210|                        WI - Milwaukee|
        14|              $10514.28|              $5562.50|                 $4522.78|
+--------------------+----------+--------------+----------------------+------------
+-------------+----------------+-----------------------------------+---------------
+----------------------+---------------------+------------------------+
```

Ahora elimine los dos registros mal formados, tal y como se indica a continuación:

```
medicare_dataframe = medicare_res.toDF()
medicare_dataframe = medicare_dataframe.where("'provider id' is NOT NULL")
```

Paso 4: Asignar los datos y utilizar las funciones Lambda de Apache Spark

AWS Glue todavía no es compatible directamente con funciones Lambda, también conocidas como funciones definidas por el usuario. Pero siempre puede convertir un elemento `DynamicFrame` en un elemento `DataFrame` de Apache Spark y viceversa, para aprovechar la funcionalidad de Spark, además de las características especiales de `DynamicFrames`.

A continuación, convierta la información de pago en números, para que los motores de análisis como Amazon Redshift o Amazon Athena puedan controlar más deprisa sus números:

```
from pyspark.sql.functions import udf
from pyspark.sql.types import StringType

chop_f = udf(lambda x: x[1:], StringType())
medicare_dataframe = medicare_dataframe.withColumn(
        "ACC", chop_f(
            medicare_dataframe["average covered charges"])).withColumn(
```

```
                "ATP", chop_f(
                    medicare_dataframe["average total payments"])).withColumn(
                        "AMP", chop_f(
                            medicare_dataframe["average medicare payments"]))
medicare_dataframe.select(['ACC', 'ATP', 'AMP']).show()
```

La salida de la llamada show es la siguiente:

```
+--------+-------+-------+
|     ACC|    ATP|    AMP|
+--------+-------+-------+
|32963.07|5777.24|4763.73|
|15131.85|5787.57|4976.71|
|37560.37|5434.95|4453.79|
|13998.28|5417.56|4129.16|
|31633.27|5658.33|4851.44|
|16920.79|6653.80|5374.14|
|11977.13|5834.74|4761.41|
|35841.09|8031.12|5858.50|
|28523.39|6113.38|5228.40|
|75233.38|5541.05|4386.94|
|67327.92|5461.57|4493.57|
|39607.28|5356.28|4408.20|
|22862.23|5374.65|4186.02|
|31110.85|5366.23|4376.23|
|25411.33|5282.93|4383.73|
| 9234.51|5676.55|4509.11|
|15895.85|5930.11|3972.85|
|19721.16|6192.54|5179.38|
|10710.88|4968.00|3898.88|
|51343.75|5996.00|4962.45|
+--------+-------+-------+
only showing top 20 rows
```

Sigue habiendo cadenas en los datos. Podemos utilizar el potente método de transformación apply_mapping para rechazar, renombrar, difundir y anidar los datos para que otros lenguajes de programación y sistemas de datos puedan obtener fácilmente acceso a ellos:

```
from awsglue.dynamicframe import DynamicFrame
medicare_tmp_dyf = DynamicFrame.fromDF(medicare_dataframe, glueContext, "nested")
medicare_nest_dyf = medicare_tmp_dyf.apply_mapping([('drg definition', 'string', 'drg',
 'string'),
```

```
                ('provider id', 'long', 'provider.id', 'long'),
                ('provider name', 'string', 'provider.name', 'string'),
                ('provider city', 'string', 'provider.city', 'string'),
                ('provider state', 'string', 'provider.state', 'string'),
                ('provider zip code', 'long', 'provider.zip', 'long'),
                ('hospital referral region description', 'string','rr', 'string'),
                ('ACC', 'string', 'charges.covered', 'double'),
                ('ATP', 'string', 'charges.total_pay', 'double'),
                ('AMP', 'string', 'charges.medicare_pay', 'double')])
medicare_nest_dyf.printSchema()
```

La salida de `printSchema` es la siguiente:

```
root
 |-- drg: string
 |-- provider: struct
 |    |-- id: long
 |    |-- name: string
 |    |-- city: string
 |    |-- state: string
 |    |-- zip: long
 |-- rr: string
 |-- charges: struct
 |    |-- covered: double
 |    |-- total_pay: double
 |    |-- medicare_pay: double
```

Si vuelve a convertir los datos en un elemento `DataFrame` de Spark, puede mostrar cómo son ahora:

```
medicare_nest_dyf.toDF().show()
```

La salida es la siguiente:

```
+--------------------+--------------------+--------------+--------------------+
|                 drg|            provider|            rr|             charges|
+--------------------+--------------------+--------------+--------------------+
|039 - EXTRACRANIA...|[10001,SOUTHEAST ...|    AL - Dothan|[32963.07,5777.24...|
|039 - EXTRACRANIA...|[10005,MARSHALL M...|AL - Birmingham|[15131.85,5787.57...|
|039 - EXTRACRANIA...|[10006,ELIZA COFF...|AL - Birmingham|[37560.37,5434.95...|
|039 - EXTRACRANIA...|[10011,ST VINCENT...|AL - Birmingham|[13998.28,5417.56...|
|039 - EXTRACRANIA...|[10016,SHELBY BAP...|AL - Birmingham|[31633.27,5658.33...|
```

```
|039 - EXTRACRANIA...|[10023,BAPTIST ME...|AL - Montgomery|[16920.79,6653.8,...|
|039 - EXTRACRANIA...|[10029,EAST ALABA...|AL - Birmingham|[11977.13,5834.74...|
|039 - EXTRACRANIA...|[10033,UNIVERSITY...|AL - Birmingham|[35841.09,8031.12...|
|039 - EXTRACRANIA...|[10039,HUNTSVILLE...|AL - Huntsville|[28523.39,6113.38...|
|039 - EXTRACRANIA...|[10040,GADSDEN RE...|AL - Birmingham|[75233.38,5541.05...|
|039 - EXTRACRANIA...|[10046,RIVERVIEW ...|AL - Birmingham|[67327.92,5461.57...|
|039 - EXTRACRANIA...|[10055,FLOWERS HO...|    AL - Dothan|[39607.28,5356.28...|
|039 - EXTRACRANIA...|[10056,ST VINCENT...|AL - Birmingham|[22862.23,5374.65...|
|039 - EXTRACRANIA...|[10078,NORTHEAST ...|AL - Birmingham|[31110.85,5366.23...|
|039 - EXTRACRANIA...|[10083,SOUTH BALD...|    AL - Mobile|[25411.33,5282.93...|
|039 - EXTRACRANIA...|[10085,DECATUR GE...|AL - Huntsville|[9234.51,5676.55,...|
|039 - EXTRACRANIA...|[10090,PROVIDENCE...|    AL - Mobile|[15895.85,5930.11...|
|039 - EXTRACRANIA...|[10092,D C H REGI...|AL - Tuscaloosa|[19721.16,6192.54...|
|039 - EXTRACRANIA...|[10100,THOMAS HOS...|    AL - Mobile|[10710.88,4968.0,...|
|039 - EXTRACRANIA...|[10103,BAPTIST ME...|AL - Birmingham|[51343.75,5996.0,...|
+--------------------+--------------------+---------------+--------------------+
only showing top 20 rows
```

Paso 5: Escribir los datos en Apache Parquet

AWS Glue le facilita la tarea de escribir los datos en un formato como Apache Parquet que las bases de datos relacionales pueden utilizar de manera eficaz:

```
glueContext.write_dynamic_frame.from_options(
       frame = medicare_nest_dyf,
       connection_type = "s3",
       connection_options = {"path": "s3://glue-sample-target/output-dir/
medicare_parquet"},
       format = "parquet")
```

Referencia de las extensiones de PySpark de AWS Glue

AWS Glue ha creado las siguientes extensiones del dialecto PySpark Python.

- Acceso a los parámetros mediante `getResolvedOptions`
- Tipos de extensión PySpark
- Clase DynamicFrame
- Clase DynamicFrameCollection
- Clase DynamicFrameWriter
- DynamicFrameReader clase

- GlueContext clase

Acceso a los parámetros mediante **`getResolvedOptions`**

La función de utilidad `getResolvedOptions(args, options)` de AWS Glue le da acceso a los argumentos que se pasan a su script cuando ejecuta un flujo de trabajo. Para utilizar esta función, comience importándola desde el módulo `utils` de AWS Glue junto con el módulo `sys`:

```
import sys
from awsglue.utils import getResolvedOptions
```

`getResolvedOptions(args, options)`

- `args`: lista de argumentos contenida en `sys.argv`.
- `options`: matriz de Python de los nombres de argumentos que quiere recuperar.

Example Recuperación de los argumentos que se pasan a un JobRun

Imagine que ha creado un JobRun en un script, quizás dentro de una función Lambda:

```
response = client.start_job_run(
             JobName = 'my_test_Job',
             Arguments = {
               '--day_partition_key':   'partition_0',
               '--hour_partition_key':  'partition_1',
               '--day_partition_value':  day_partition_value,
               '--hour_partition_value': hour_partition_value } )
```

Para recuperar los argumentos que se pasan, puede utilizar la función `getResolvedOptions` de la siguiente manera:

```
import sys
from awsglue.utils import getResolvedOptions

args = getResolvedOptions(sys.argv,
                          ['JOB_NAME',
                           'day_partition_key',
                           'hour_partition_key',
                           'day_partition_value',
                           'hour_partition_value'])
```

```
print "The day-partition key is: ", args['day_partition_key']
print "and the day-partition value is: ", args['day_partition_value']
```

Tenga en cuenta que al definir cada uno de los argumentos comienzan con dos guiones y después se hace referencia a ellos en el script sin los guiones. Los argumentos solo usan guiones bajos, no guiones. Sus argumentos tienen que seguir esta convención para que se puedan resolver.

Tipos de extensión PySpark

Los tipos usados por las extensiones PySpark de AWS Glue.

DataType

La clase base para los otros tipos de AWS Glue.

`__init__(properties={})`

- `properties`: propiedades del tipo de datos (opcional).

`typeName(cls)`

Devuelve el tipo de la clase de tipo de AWS Glue (es decir, el nombre de la clase con "Type" eliminado del final).

- `cls`: una instancia de clase de AWS Glue derivada de `DataType`.

`jsonValue( )`

Devuelve un objeto JSON que contiene las propiedades y los tipos de datos de la clase:

```
{
  "dataType": typeName,
  "properties": properties
}
```

AtomicType y derivados sencillos

Se hereda de la clase DataType y la amplía, además de servir como clase base para todos los tipos de datos atómicos de AWS Glue.

`fromJsonValue(cls, json_value)`

Inicializa una instancia de la clase con valores a partir de un objeto JSON.

- `cls`: una instancia de la clase de tipo de AWS Glue que se va a inicializar.
- `json_value`: objeto JSON desde el que se van a cargar pares clave-valor.

Los siguientes tipos son derivados sencillos de la clase AtomicType:

- `BinaryType` – Datos binarios.
- `BooleanType`: valores booleanos.
- `ByteType`: un valor de byte.
- `DateType`: un valor datetime.
- `DoubleType`: un valor doble de punto flotante.
- `IntegerType` – Un valor entero.
- `LongType`: un valor entero largo.
- `NullType`: un valor nulo.
- `ShortType`: un valor entero corto.
- `StringType`: una cadena de texto.
- `TimestampType`: un valor de marca temporal (normalmente en segundos desde el 01/01/1970).
- `UnknownType`: un valor de tipo no identificado.

DecimalType(AtomicType)

Se hereda de la clase AtomicType y la amplía para representar un número decimal (un número expresado en dígitos decimales, en contraposición a los números de base 2 binarios).

`__init__(precision=10, scale=2, properties={})`

- `precision`: el número de dígitos en el número decimal (opcional; el valor predeterminado es 10).
- `scale`: el número de dígitos situados a la derecha del punto decimal (opcional; el valor predeterminado es 2).
- `properties`: las propiedades del número decimal (opcional).

EnumType(AtomicType)

Se hereda de la clase AtomicType y la amplía para representar una enumeración de opciones válidas.

`__init__(options)`

- `options`: una lista de las opciones que se enumeran.

tipos de colección

- ArrayType(DataType)
- ChoiceType(DataType)
- MapType(DataType)
- Field(Object)
- StructType(DataType)
- (EntityType DataType)

ArrayType(DataType)

`__init__(elementType=UnknownType(), properties={})`

- `elementType`: el tipo de elementos en la matriz (opcional; el valor predeterminado es UnknownType).
- `properties`: propiedades de la matriz (opcional).

ChoiceType(DataType)

`__init__(choices=[], properties={})`

- `choices`: una lista de posibles opciones (opcional).
- `properties`: propiedades de estas opciones (opcional).

`add(new_choice)`

Añade una nueva opción a la lista de posibles opciones.

- `new_choice`: la opción que se va a añadir a la lista de posibles opciones.

`merge(new_choices)`

Combina una lista de nuevas opciones con la lista de opciones existente.

- `new_choices`: una lista de nuevas opciones que se van a combinar con las opciones existentes.

MapType(DataType)

`__init__(valueType=UnknownType, properties={})`

- `valueType`: el tipo de valores en el mapa (opcional; el valor predeterminado es UnknownType).
- `properties`: propiedades del mapa (opcional).

Field(Object)

Crea un objeto de campo fuera de un objeto que deriva de DataType.

`__init__(name, dataType, properties={})`

- `name`: el nombre que se va a asignar al campo.
- `dataType`: el objeto a partir del cual se va a crear un campo.
- `properties`: propiedades del campo (opcional).

StructType(DataType)

Define una estructura de datos (`struct`).

`__init__(fields=[], properties={})`

- `fields`: una lista de los campos (de tipo `Field`) que se va a incluir en la estructura (opcional).
- `properties`: propiedades de la estructura (opcional).

`add(field)`

- `field`: un objeto de tipo `Field` para añadir a la estructura.

`hasField(field)`

Devuelve `True` si esta estructura tiene un campo del mismo nombre, o `False` si no.

- `field`: un nombre de campo o un objeto de tipo `Field` cuyo nombre se usa.

`getField(field)`

- `field`: un nombre de campo o un objeto de tipo `Field` cuyo nombre se usa. Si la estructura tiene un campo del mismo nombre, se devuelve.

(EntityType DataType)

`__init__(entity, base_type, properties)`

Esta clase no se ha implementado aún.

otros tipos

- DataSource(object)
- DataSink(object)

DataSource(object)

`__init__(j_source, sql_ctx, name)`

- `j_source`: el origen de datos.
- `sql_ctx`: el contexto SQL.
- `name`: el nombre de origen de datos.

`setFormat(format, **options)`

- `format`: el formato que se va a establecer para el origen de datos.
- `options`: un conjunto de opciones que se va a establecer para el origen de datos. Para obtener más información sobre las opciones de formato, consulte the section called “Opciones de formato de datos”.

`getFrame()`

Devuelve un `DynamicFrame` para el origen de datos.

DataSink(object)

`__init__(j_sink, sql_ctx)`

- `j_sink`: el receptor que se va a crear.
- `sql_ctx`: el contexto SQL para el receptor de datos.

`setFormat(format, **options)`

- `format`: el formato que se va a establecer para el receptor de datos.
- `options`: un conjunto de opciones que se va a establecer para el receptor de datos. Para obtener más información sobre las opciones de formato, consulte the section called “Opciones de formato de datos”.

`setAccumulableSize(size)`

- `size`: el tamaño acumulable que se va a establecer, en bytes.

`writeFrame(dynamic_frame, info="")`

- `dynamic_frame`: el `DynamicFrame` que se va a escribir.
- `info`: información acerca del `DynamicFrame` (opcional).

`write(dynamic_frame_or_dfc, info="")`

Escribe un `DynamicFrame` o `DynamicFrameCollection`.

- `dynamic_frame_or_dfc`: un objeto `DynamicFrame` o `DynamicFrameCollection` que se van a escribir.
- `info`: información acerca de los `DynamicFrame` o `DynamicFrames` que se van a escribir (opcional).

Clase DynamicFrame

Una de las principales abstracciones de Apache Spark es el `DataFrame` de SparkSQL, que es similar a la construcción `DataFrame` que se encuentra en R y en Pandas. Un elemento `DataFrame` es similar a una tabla y admite operaciones de estilo funcional (map/reduce/filter/etc.) y operaciones SQL (select, project, aggregate).

`DataFrames` son eficaces y de uso general, pero tienen limitaciones en las operaciones de extracción, transformación y carga (ETL). Más aún, es preciso especificar un esquema antes de cargar cualquier dato. SparkSQL aborda esta cuestión al repasar dos veces los datos: la primera para deducir el esquema y la segunda para cargar los datos. Sin embargo, esta deducción es limitada y no se ocupa de la realidad de los datos confusos. Por ejemplo, el mismo campo puede ser de un tipo diferente en diferentes registros. Apache Spark a menudo renuncia e informa del tipo como si fuera una `string` mediante el texto del campo original. Esto podría no ser correcto y puede que desee controlar mejor cómo se resuelven las discrepancias en el esquema. Y en cuanto a grandes conjuntos de datos, un paso adicional por los datos de origen podría tener un precio prohibitivo.

Para abordar estas limitaciones, AWS Glue introduce `DynamicFrame`. `DynamicFrame` es similar a `DataFrame`, salvo que cada registro se describe a sí mismo, por lo que no es preciso tener un esquema inicial. En su lugar, AWS Glue procesa un esquema sobre la marcha cuando es necesario y codifica explícitamente las incoherencias de esquema mediante un tipo de elección (o unión). Puede resolver estas incoherencias para que los conjuntos de datos pasen a ser compatibles con almacenes de datos que requieren un esquema fijo.

Asimismo, un `DynamicRecord` representa un registro lógico dentro de un `DynamicFrame`. Es como una fila en un `DataFrame` de Spark, salvo que es autodescriptivo y se puede utilizar para datos que no se ajustan a un esquema fijo. Cuando se utiliza Glue AWS con PySpark, no suele manipular de forma independiente `DynamicRecords`. Más bien, transformará el conjunto de datos en conjunto a través de `DynamicFrame`.

Puede convertir `DynamicFrames` en `DataFrames` y viceversa, una vez que haya resuelto las incoherencias de esquema.

— construcción —

- __init__
- fromDF
- toDF

__init__

`__init__(jdf, glue_ctx, name)`

- `jdf`: referencia a la trama de datos de la máquina virtual de Java (JVM).
- `glue_ctx`: un objeto GlueContext clase.
- `name`: cadena de nombre opcional, vacía de forma predeterminada.

fromDF

`fromDF(dataframe, glue_ctx, name)`

Convierte un `DataFrame` en un `DynamicFrame` al convertir campos de `DataFrame` en campos de `DynamicRecord`. Devuelve el nuevo `DynamicFrame`.

Un `DynamicRecord` representa un registro lógico en un `DynamicFrame`. Es similar a una fila en un `DataFrame` de Spark, salvo que es autodescriptivo y se puede utilizar para datos que no se ajustan a un esquema fijo.

Esta función espera que las columnas con nombres duplicados en el `DataFrame` ya se hayan resuelto.

- `dataframe`: `DataFrame` de Apache Spark SQL que debe convertirse (obligatorio).
- `glue_ctx`: objeto GlueContext clase que especifica el contexto de esta transformación (obligatorio).
- `name`: el nombre de `DynamicFrame` que resulte (opcional desde AWS Glue 3.0).

toDF

`toDF(options)`

Convierte un `DynamicFrame` en un `DataFrame` de Apache Spark convirtiendo campos de `DynamicRecords` en campos de `DataFrame`. Devuelve el nuevo `DataFrame`.

Un `DynamicRecord` representa un registro lógico en un `DynamicFrame`. Es similar a una fila en un `DataFrame` de Spark, salvo que es autodescriptivo y se puede utilizar para datos que no se ajustan a un esquema fijo.

- `options`: lista de opciones. Especifique el tipo de destino si ha elegido el tipo de acción `Project` y `Cast`. Algunos ejemplos son los siguientes:

```
>>>toDF([ResolveOption("a.b.c", "KeepAsStruct")])
>>>toDF([ResolveOption("a.b.c", "Project", DoubleType())])
```

— información —

- count
- Esquema
- printSchema
- show
- repartition
- coalesce

count

`count( )`: devuelve el número de filas del `DataFrame` subyacente.

Esquema

`schema( )` - Muestra el esquema de este `DynamicFrame` o en caso de que no esté disponible, el esquema del elemento `DataFrame` subyacente.

Para obtener más información acerca de los tipos de `DynamicFrame` que componen este esquema, consulte the section called “Tipos”.

printSchema

`printSchema( )`: imprime el esquema del `DataFrame` subyacente.

show

`show(num_rows)`: imprime un número especificado de filas del `DataFrame` subyacente.

repartition

`repartition(numPartitions)`: devuelve un `DynamicFrame` nuevo con particiones `numPartitions`.

coalesce

`coalesce(numPartitions)`: devuelve un `DynamicFrame` nuevo con particiones `numPartitions`.

— transformaciones —

- apply_mapping
- drop_fields
- filtro
- unirse
- map
- mergeDynamicFrame
- relationalize
- rename_field
- resolveChoice
- select_fields
- spigot
- split_fields
- split_rows
- unbox
- the section called “unión”
- unnest
- unnest_ddb_json
- write

apply_mapping

`apply_mapping(mappings, transformation_ctx="", info="", stageThreshold=0, totalThreshold=0)`

Aplica una asignación declarativa a un `DynamicFrame` y devuelve un `DynamicFrame` nuevo con esas asignaciones aplicadas a los campos que se especifican. Los campos no especificados se omiten en el `DynamicFrame` nuevo.

- `mappings`: una lista de tuplas de asignación (obligatorio). Cada una consta de (columna de origen, tipo de origen, columna de destino, tipo de destino).

  Si la columna de origen tiene un punto "`.`" en el nombre, se deben colocar comillas simples "```" antes y después. Por ejemplo, para asignar `this.old.name` (cadena) a `thisNewName`, debe utilizar las siguientes tuplas:

  ```
  ("`this.old.name`", "string", "thisNewName", "string")
  ```

- `transformation_ctx`: cadena única que se utiliza para identificar la información del estado (opcional).
- `info`: cadena que se asociará a la notificación de errores para esta transformación (opcional).
- `stageThreshold`: cantidad de errores detectados durante esta transformación que provocarán que el proceso falle (opcional). El valor predeterminado es cero, lo que indica que el proceso no debería fallar.
- `totalThreshold`: cantidad de errores detectados hasta esta transformación incluida que provocarán que el proceso falle (opcional). El valor predeterminado es cero, lo que indica que el proceso no debería fallar.

Ejemplo: uso de apply_mapping para cambiar el nombre de los campos y cambiar los tipos de campos

En el siguiente ejemplo de código, se muestra cómo utilizar el método `apply_mapping` para cambiar el nombre de los campos seleccionados y cambiar los tipos de campos.

> **Note**
>
> Para acceder al conjunto de datos utilizado en este ejemplo, consulte Código de ejemplo: unión de los datos y establecimiento de relaciones entre ellos y siga las instrucciones en Paso 1: Rastrear los datos del bucket de Amazon S3.

```
# Example: Use apply_mapping to reshape source data into
# the desired column names and types as a new DynamicFrame

from pyspark.context import SparkContext
from awsglue.context import GlueContext
```

```
# Create GlueContext
sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

# Create a DynamicFrame and view its schema
persons = glueContext.create_dynamic_frame.from_catalog(
    database="legislators", table_name="persons_json"
)
print("Schema for the persons DynamicFrame:")
persons.printSchema()

# Select and rename fields, change field type
print("Schema for the persons_mapped DynamicFrame, created with apply_mapping:")
persons_mapped = persons.apply_mapping(
    [
        ("family_name", "String", "last_name", "String"),
        ("name", "String", "first_name", "String"),
        ("birth_date", "String", "date_of_birth", "Date"),
    ]
)
persons_mapped.printSchema()
```

Salida

```
Schema for the persons DynamicFrame:
root
|-- family_name: string
|-- name: string
|-- links: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- url: string
|-- gender: string
|-- image: string
|-- identifiers: array
|    |-- element: struct
|    |    |-- scheme: string
|    |    |-- identifier: string
|-- other_names: array
|    |-- element: struct
|    |    |-- lang: string
|    |    |-- note: string
```

```
|    |    |-- name: string
|-- sort_name: string
|-- images: array
|    |-- element: struct
|    |    |-- url: string
|-- given_name: string
|-- birth_date: string
|-- id: string
|-- contact_details: array
|    |-- element: struct
|    |    |-- type: string
|    |    |-- value: string
|-- death_date: string

Schema for the persons_mapped DynamicFrame, created with apply_mapping:
root
|-- last_name: string
|-- first_name: string
|-- date_of_birth: date
```

drop_fields

`drop_fields(paths, transformation_ctx="", info="", stageThreshold=0, totalThreshold=0)`

Llama a la transformación Clase FlatMap para eliminar campos de un elemento `DynamicFrame`. Devuelve un `DynamicFrame` nuevo con los campos especificados anulados.

- `paths`: una lista de cadenas. Cada una contiene la ruta completa a un nodo de campo que desea descartar. Puede utilizar la notación de puntos para especificar campos anidados. Por ejemplo, si el campo `first` es un elemento secundario del campo `name` en el árbol, especificará `"name.first"` para la ruta.

  Si el nombre de un nodo de campo tiene un literal `.`, debe escribir el nombre entre tildes graves (`` ` ``).
- `transformation_ctx`: cadena única que se utiliza para identificar la información del estado (opcional).
- `info`: cadena que se asociará a la notificación de errores para esta transformación (opcional).
- `stageThreshold`: cantidad de errores detectados durante esta transformación que provocarán que el proceso falle (opcional). El valor predeterminado es cero, lo que indica que el proceso no debería fallar.

- `totalThreshold`: cantidad de errores detectados hasta esta transformación incluida que provocarán que el proceso falle (opcional). El valor predeterminado es cero, lo que indica que el proceso no debería fallar.

Ejemplo: uso de drop_fields para descartar campos de un **`DynamicFrame`**

En este ejemplo de código se utiliza el método `drop_fields` para eliminar los campos anidados y de nivel superior seleccionados de un `DynamicFrame`.

Conjunto de datos de ejemplo

El ejemplo utiliza el siguiente conjunto de datos que está representado por la tabla `EXAMPLE-FRIENDS-DATA` en el código:

```
{"name": "Sally", "age": 23, "location": {"state": "WY", "county": "Fremont"},
 "friends": []}
{"name": "Varun", "age": 34, "location": {"state": "NE", "county": "Douglas"},
 "friends": [{"name": "Arjun", "age": 3}]}
{"name": "George", "age": 52, "location": {"state": "NY"}, "friends": [{"name":
 "Fred"}, {"name": "Amy", "age": 15}]}
{"name": "Haruki", "age": 21, "location": {"state": "AK", "county": "Denali"}}
{"name": "Sheila", "age": 63, "friends": [{"name": "Nancy", "age": 22}]}
```

Código de ejemplo

```
# Example: Use drop_fields to remove top-level and nested fields from a DynamicFrame.
# Replace MY-EXAMPLE-DATABASE with your Glue Data Catalog database name.
# Replace EXAMPLE-FRIENDS-DATA with your table name.

from pyspark.context import SparkContext
from awsglue.context import GlueContext

# Create GlueContext
sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

# Create a DynamicFrame from Glue Data Catalog
glue_source_database = "MY-EXAMPLE-DATABASE"
glue_source_table = "EXAMPLE-FRIENDS-DATA"

friends = glueContext.create_dynamic_frame.from_catalog(
```

```
    database=glue_source_database, table_name=glue_source_table
)
print("Schema for friends DynamicFrame before calling drop_fields:")
friends.printSchema()

# Remove location.county, remove friends.age, remove age
friends = friends.drop_fields(paths=["age", "location.county", "friends.age"])
print("Schema for friends DynamicFrame after removing age, county, and friend age:")
friends.printSchema()
```

Salida

```
Schema for friends DynamicFrame before calling drop_fields:
root
|-- name: string
|-- age: int
|-- location: struct
|    |-- state: string
|    |-- county: string
|-- friends: array
|    |-- element: struct
|    |    |-- name: string
|    |    |-- age: int

Schema for friends DynamicFrame after removing age, county, and friend age:
root
|-- name: string
|-- location: struct
|    |-- state: string
|-- friends: array
|    |-- element: struct
|    |    |-- name: string
```

filtro

`filter(f, transformation_ctx="", info="", stageThreshold=0, totalThreshold=0)`

Devuelve un `DynamicFrame` nuevo que contiene todos los `DynamicRecords` dentro de la entrada `DynamicFrame` que cumplen la función `f` de predicado especificada.

- `f`: función de predicado que debe aplicarse al elemento `DynamicFrame`. La función debe tomar un `DynamicRecord` como argumento y devolver True si `DynamicRecord` cumple los requisitos de filtro o False si no los cumple (obligatorio).

  Un `DynamicRecord` representa un registro lógico en un `DynamicFrame`. Es similar a una fila en un `DataFrame` de Spark, salvo que es autodescriptivo y se puede utilizar para datos que no se ajustan a un esquema fijo.
- `transformation_ctx`: cadena única que se utiliza para identificar la información del estado (opcional).
- `info`: cadena que se asociará a la notificación de errores para esta transformación (opcional).
- `stageThreshold`: cantidad de errores detectados durante esta transformación que provocarán que el proceso falle (opcional). El valor predeterminado es cero, lo que indica que el proceso no debería fallar.
- `totalThreshold`: cantidad de errores detectados hasta esta transformación incluida que provocarán que el proceso falle (opcional). El valor predeterminado es cero, lo que indica que el proceso no debería fallar.

Ejemplo: uso de filter para obtener una selección filtrada de campos

En este ejemplo se utiliza el método `filter` para crear un `DynamicFrame` nuevo que incluye una selección filtrada de campos de otro `DynamicFrame`.

Al igual que el método `map`, `filter` toma una función como un argumento que se aplica a cada registro en el `DynamicFrame` original. La función toma un registro como una entrada y devuelve un valor booleano. Si el valor que se devuelve es verdadero, el registro se incluye en el `DynamicFrame` resultante. Si es falso, el registro se omite.

> **Note**
>
> Para acceder al conjunto de datos utilizado en este ejemplo, consulte Ejemplo de código: Preparación de datos con ResolveChoice, Lambda y ApplyMapping y siga las instrucciones en Paso 1: Rastrear los datos del bucket de Amazon S3.

```
# Example: Use filter to create a new DynamicFrame
# with a filtered selection of records
```

```
from pyspark.context import SparkContext
from awsglue.context import GlueContext

# Create GlueContext
sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

# Create DynamicFrame from Glue Data Catalog
medicare = glueContext.create_dynamic_frame.from_options(
    "s3",
    {
        "paths": [
            "s3://awsglue-datasets/examples/medicare/Medicare_Hospital_Provider.csv"
        ]
    },
    "csv",
    {"withHeader": True},
)

# Create filtered DynamicFrame with custom lambda
# to filter records by Provider State and Provider City
sac_or_mon = medicare.filter(
    f=lambda x: x["Provider State"] in ["CA", "AL"]
    and x["Provider City"] in ["SACRAMENTO", "MONTGOMERY"]
)

# Compare record counts
print("Unfiltered record count: ", medicare.count())
print("Filtered record count:  ", sac_or_mon.count())
```

Salida

```
Unfiltered record count:  163065
Filtered record count:   564
```

unirse

`join(paths1, paths2, frame2, transformation_ctx="", info="", stageThreshold=0, totalThreshold=0)`

Realiza una unión de igualdad con otro elemento `DynamicFrame` y devuelve el elemento `DynamicFrame` resultante.

- `paths1`: lista de las claves de esta trama que deben unirse.
- `paths2`: lista de las claves de la otra trama que deben unirse.
- `frame2`: el otro elemento `DynamicFrame` que debe unirse.
- `transformation_ctx`: cadena única que se utiliza para identificar la información del estado (opcional).
- `info`: cadena que se asociará a la notificación de errores para esta transformación (opcional).
- `stageThreshold`: cantidad de errores detectados durante esta transformación que provocarán que el proceso falle (opcional). El valor predeterminado es cero, lo que indica que el proceso no debería fallar.
- `totalThreshold`: cantidad de errores detectados hasta esta transformación incluida que provocarán que el proceso falle (opcional). El valor predeterminado es cero, lo que indica que el proceso no debería fallar.

Ejemplo: uso de join para combinar **`DynamicFrames`**

En este ejemplo se utiliza el método `join` para realizar una unión en tres `DynamicFrames`. AWS Glue realiza la unión en función de las claves de campos que se proporcionan. El `DynamicFrame` resultante contiene filas de los dos marcos originales donde coinciden las claves especificadas.

Tenga en cuenta que la transformación `join` mantiene todos los campos intactos. Esto significa que los campos que se especifican para que coincidan aparecen en el DynamicFrame resultante, incluso si son redundantes y contienen las mismas claves. En este ejemplo, utilizamos `drop_fields` para eliminar estas claves redundantes después de la unión.

> **Note**
>
> Para acceder al conjunto de datos utilizado en este ejemplo, consulte Código de ejemplo: unión de los datos y establecimiento de relaciones entre ellos y siga las instrucciones en Paso 1: Rastrear los datos del bucket de Amazon S3.

```
# Example: Use join to combine data from three DynamicFrames

from pyspark.context import SparkContext
from awsglue.context import GlueContext

# Create GlueContext
```

```
sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

# Load DynamicFrames from Glue Data Catalog
persons = glueContext.create_dynamic_frame.from_catalog(
    database="legislators", table_name="persons_json"
)
memberships = glueContext.create_dynamic_frame.from_catalog(
    database="legislators", table_name="memberships_json"
)
orgs = glueContext.create_dynamic_frame.from_catalog(
    database="legislators", table_name="organizations_json"
)
print("Schema for the persons DynamicFrame:")
persons.printSchema()
print("Schema for the memberships DynamicFrame:")
memberships.printSchema()
print("Schema for the orgs DynamicFrame:")
orgs.printSchema()

# Join persons and memberships by ID
persons_memberships = persons.join(
    paths1=["id"], paths2=["person_id"], frame2=memberships
)

# Rename and drop fields from orgs
# to prevent field name collisions with persons_memberships
orgs = (
    orgs.drop_fields(["other_names", "identifiers"])
    .rename_field("id", "org_id")
    .rename_field("name", "org_name")
)

# Create final join of all three DynamicFrames
legislators_combined = orgs.join(
    paths1=["org_id"], paths2=["organization_id"], frame2=persons_memberships
).drop_fields(["person_id", "org_id"])

# Inspect the schema for the joined data
print("Schema for the new legislators_combined DynamicFrame:")
legislators_combined.printSchema()
```

Salida

```
Schema for the persons DynamicFrame:
root
|-- family_name: string
|-- name: string
|-- links: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- url: string
|-- gender: string
|-- image: string
|-- identifiers: array
|    |-- element: struct
|    |    |-- scheme: string
|    |    |-- identifier: string
|-- other_names: array
|    |-- element: struct
|    |    |-- lang: string
|    |    |-- note: string
|    |    |-- name: string
|-- sort_name: string
|-- images: array
|    |-- element: struct
|    |    |-- url: string
|-- given_name: string
|-- birth_date: string
|-- id: string
|-- contact_details: array
|    |-- element: struct
|    |    |-- type: string
|    |    |-- value: string
|-- death_date: string

Schema for the memberships DynamicFrame:
root
|-- area_id: string
|-- on_behalf_of_id: string
|-- organization_id: string
|-- role: string
|-- person_id: string
|-- legislative_period_id: string
|-- start_date: string
|-- end_date: string
```

```
Schema for the orgs DynamicFrame:
root
|-- identifiers: array
|    |-- element: struct
|    |    |-- scheme: string
|    |    |-- identifier: string
|-- other_names: array
|    |-- element: struct
|    |    |-- lang: string
|    |    |-- note: string
|    |    |-- name: string
|-- id: string
|-- classification: string
|-- name: string
|-- links: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- url: string
|-- image: string
|-- seats: int
|-- type: string

Schema for the new legislators_combined DynamicFrame:
root
|-- role: string
|-- seats: int
|-- org_name: string
|-- links: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- url: string
|-- type: string
|-- sort_name: string
|-- area_id: string
|-- images: array
|    |-- element: struct
|    |    |-- url: string
|-- on_behalf_of_id: string
|-- other_names: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- name: string
|    |    |-- lang: string
```

```
|-- contact_details: array
|    |-- element: struct
|    |    |-- type: string
|    |    |-- value: string
|-- name: string
|-- birth_date: string
|-- organization_id: string
|-- gender: string
|-- classification: string
|-- legislative_period_id: string
|-- identifiers: array
|    |-- element: struct
|    |    |-- scheme: string
|    |    |-- identifier: string
|-- image: string
|-- given_name: string
|-- start_date: string
|-- family_name: string
|-- id: string
|-- death_date: string
|-- end_date: string
```

map

`map(f, transformation_ctx="", info="", stageThreshold=0, totalThreshold=0)`

Devuelve un elemento `DynamicFrame` nuevo que se obtiene al aplicar la función de mapeo especificada a todos los registros del elemento `DynamicFrame` original.

- `f`: función de asignación que debe aplicarse a todos los registros del elemento `DynamicFrame`. La función debe tomar un elemento `DynamicRecord` como argumento y devolver un `DynamicRecord` nuevo (obligatorio).

  Un `DynamicRecord` representa un registro lógico en un `DynamicFrame`. Es similar a una fila en un `DataFrame` de Apache Spark, salvo que es autodescriptivo y se puede utilizar para datos que no se ajustan a un esquema fijo.
- `transformation_ctx`: cadena única que se utiliza para identificar la información del estado (opcional).
- `info`: cadena que está asociada con errores en la transformación (opcional).
- `stageThreshold`: cantidad máxima de errores que se puede producir en la transformación antes de que se determine que es errónea (opcional). El rol predeterminado es cero.

- `totalThreshold`: cantidad máxima de errores que se pueden producir en total antes de que se determine que el proceso es erróneo (opcional). El rol predeterminado es cero.

Ejemplo: uso de map para aplicar una función a cada registro en un **`DynamicFrame`**

En este ejemplo se muestra cómo utilizar el método `map` para aplicar una función a cada registro de un `DynamicFrame`. En concreto, este ejemplo aplica una función llamada `MergeAddress` a cada registro para combinar varios campos de dirección en un solo tipo de `struct`.

> **Note**
>
> Para acceder al conjunto de datos utilizado en este ejemplo, consulte Ejemplo de código: Preparación de datos con ResolveChoice, Lambda y ApplyMapping y siga las instrucciones en Paso 1: Rastrear los datos del bucket de Amazon S3.

```
# Example: Use map to combine fields in all records
# of a DynamicFrame

from pyspark.context import SparkContext
from awsglue.context import GlueContext

# Create GlueContext
sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

# Create a DynamicFrame and view its schema
medicare = glueContext.create_dynamic_frame.from_options(
        "s3",
        {"paths": ["s3://awsglue-datasets/examples/medicare/
Medicare_Hospital_Provider.csv"]},
        "csv",
        {"withHeader": True})
print("Schema for medicare DynamicFrame:")
medicare.printSchema()

# Define a function to supply to the map transform
# that merges address fields into a single field
def MergeAddress(rec):
  rec["Address"] = {}
  rec["Address"]["Street"] = rec["Provider Street Address"]
```

```
  rec["Address"]["City"] = rec["Provider City"]
  rec["Address"]["State"] = rec["Provider State"]
  rec["Address"]["Zip.Code"] = rec["Provider Zip Code"]
  rec["Address"]["Array"] = [rec["Provider Street Address"], rec["Provider City"],
 rec["Provider State"], rec["Provider Zip Code"]]
  del rec["Provider Street Address"]
  del rec["Provider City"]
  del rec["Provider State"]
  del rec["Provider Zip Code"]
  return rec


# Use map to apply MergeAddress to every record
mapped_medicare = medicare.map(f = MergeAddress)
print("Schema for mapped_medicare DynamicFrame:")
mapped_medicare.printSchema()
```

Salida

```
Schema for medicare DynamicFrame:
root
|-- DRG Definition: string
|-- Provider Id: string
|-- Provider Name: string
|-- Provider Street Address: string
|-- Provider City: string
|-- Provider State: string
|-- Provider Zip Code: string
|-- Hospital Referral Region Description: string
|-- Total Discharges: string
|-- Average Covered Charges: string
|-- Average Total Payments: string
|-- Average Medicare Payments: string

Schema for mapped_medicare DynamicFrame:
root
|-- Average Total Payments: string
|-- Average Covered Charges: string
|-- DRG Definition: string
|-- Average Medicare Payments: string
|-- Hospital Referral Region Description: string
|-- Address: struct
|    |-- Zip.Code: string
```

```
|    |-- City: string
|    |-- Array: array
|    |    |-- element: string
|    |-- State: string
|    |-- Street: string
|-- Provider Id: string
|-- Total Discharges: string
|-- Provider Name: string
```

mergeDynamicFrame

`mergeDynamicFrame(stage_dynamic_frame, primary_keys, transformation_ctx = "", options = {}, info = "", stageThreshold = 0, totalThreshold = 0)`

Combina este objeto `DynamicFrame` con una instancia provisional de `DynamicFrame` en función de las claves principales especificadas para identificar registros. Los registros duplicados (registros con las mismas claves principales) no se desduplican. Si no hay ningún registro que coincida en el marco provisional, se retienen todos los registros del origen (incluidos los duplicados). Si el marco provisional tiene registros coincidentes, estos sobrescriben a los registros del origen en AWS Glue.

- `stage_dynamic_frame`: `DynamicFrame` provisional que se fusionará.
- `primary_keys`: la lista de campos de clave principal para hacer coincidir los registros de los marcos dinámicos de origen y provisionales.
- `transformation_ctx`: cadena única que se utiliza para recuperar metadatos sobre la transformación actual (opcional).
- `options`: una cadena de pares de nombre-valor de JSON que proporcionan información adicional para esta transformación. Este argumento no se utiliza actualmente.
- `info`: un elemento `String`. Cualquier cadena que se va a asociar con los errores de esta transformación.
- `stageThreshold`: un elemento `Long`. Número de errores de la transformación especificada que provocarán que el proceso se termine.
- `totalThreshold`: un elemento `Long`. Número total de errores hasta esta transformación (incluida) que provocarán que el proceso se termine.

Este método devuelve un nuevo objeto `DynamicFrame` que se obtiene combinando esta instancia de `DynamicFrame` con la instancia de `DynamicFrame` provisional.

El objeto `DynamicFrame` devuelto contiene el registro A en estos casos:

- Si A existe tanto en el marco de origen como en el marco provisional, se devuelve A en el marco provisional.
- Si A está en la tabla de origen y `A.primaryKeys` no está en `stagingDynamicFrame` A no se actualiza en la tabla provisional.

No es necesario que el marco de origen y el marco provisional tengan el mismo esquema.

Ejemplo: utilice mergeDynamicFrame para combinar dos **`DynamicFrames`** en función de una clave principal

El siguiente ejemplo de código muestra cómo utilizar el método `mergeDynamicFrame` para combinar un `DynamicFrame` con un `DynamicFrame` de “puesta en escena”, en función del `id` de la clave principal.

Conjunto de datos de ejemplo

El ejemplo usa dos `DynamicFrames` de una `DynamicFrameCollection` llamada `split_rows_collection`. A continuación se ofrece una lista de las claves de `split_rows_collection`.

```
dict_keys(['high', 'low'])
```

Código de ejemplo

```
# Example: Use mergeDynamicFrame to merge DynamicFrames
# based on a set of specified primary keys

from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.transforms import SelectFromCollection

# Inspect the original DynamicFrames
frame_low = SelectFromCollection.apply(dfc=split_rows_collection, key="low")
print("Inspect the DynamicFrame that contains rows where ID < 10")
frame_low.toDF().show()

frame_high = SelectFromCollection.apply(dfc=split_rows_collection, key="high")
print("Inspect the DynamicFrame that contains rows where ID > 10")
frame_high.toDF().show()
```

```
# Merge the DynamicFrames based on the "id" primary key
merged_high_low = frame_high.mergeDynamicFrame(
    stage_dynamic_frame=frame_low, primary_keys=["id"]
)

# View the results where the ID is 1 or 20
print("Inspect the merged DynamicFrame that contains the combined rows")
merged_high_low.toDF().where("id = 1 or id= 20").orderBy("id").show()
```

Salida

```
Inspect the DynamicFrame that contains rows where ID < 10
+---+-----+----------------------+-----------------------+
| id|index|contact_details.val.type|contact_details.val.value|
+---+-----+----------------------+-----------------------+
|  1|    0|                   fax|           202-225-3307|
|  1|    1|                 phone|           202-225-5731|
|  2|    0|                   fax|           202-225-3307|
|  2|    1|                 phone|           202-225-5731|
|  3|    0|                   fax|           202-225-3307|
|  3|    1|                 phone|           202-225-5731|
|  4|    0|                   fax|           202-225-3307|
|  4|    1|                 phone|           202-225-5731|
|  5|    0|                   fax|           202-225-3307|
|  5|    1|                 phone|           202-225-5731|
|  6|    0|                   fax|           202-225-3307|
|  6|    1|                 phone|           202-225-5731|
|  7|    0|                   fax|           202-225-3307|
|  7|    1|                 phone|           202-225-5731|
|  8|    0|                   fax|           202-225-3307|
|  8|    1|                 phone|           202-225-5731|
|  9|    0|                   fax|           202-225-3307|
|  9|    1|                 phone|           202-225-5731|
| 10|    0|                   fax|           202-225-6328|
| 10|    1|                 phone|           202-225-4576|
+---+-----+----------------------+-----------------------+
only showing top 20 rows

Inspect the DynamicFrame that contains rows where ID > 10
+---+-----+----------------------+-----------------------+
| id|index|contact_details.val.type|contact_details.val.value|
+---+-----+----------------------+-----------------------+
| 11|    0|                   fax|           202-225-6328|
```

```
| 11|    1|                  phone|             202-225-4576|
| 11|    2|                twitter|           RepTrentFranks|
| 12|    0|                    fax|             202-225-6328|
| 12|    1|                  phone|             202-225-4576|
| 12|    2|                twitter|           RepTrentFranks|
| 13|    0|                    fax|             202-225-6328|
| 13|    1|                  phone|             202-225-4576|
| 13|    2|                twitter|           RepTrentFranks|
| 14|    0|                    fax|             202-225-6328|
| 14|    1|                  phone|             202-225-4576|
| 14|    2|                twitter|           RepTrentFranks|
| 15|    0|                    fax|             202-225-6328|
| 15|    1|                  phone|             202-225-4576|
| 15|    2|                twitter|           RepTrentFranks|
| 16|    0|                    fax|             202-225-6328|
| 16|    1|                  phone|             202-225-4576|
| 16|    2|                twitter|           RepTrentFranks|
| 17|    0|                    fax|             202-225-6328|
| 17|    1|                  phone|             202-225-4576|
+---+-----+----------------------+-------------------------+
only showing top 20 rows

Inspect the merged DynamicFrame that contains the combined rows
+---+-----+----------------------+-------------------------+
| id|index|contact_details.val.type|contact_details.val.value|
+---+-----+----------------------+-------------------------+
|  1|    0|                   fax|             202-225-3307|
|  1|    1|                 phone|             202-225-5731|
| 20|    0|                   fax|             202-225-5604|
| 20|    1|                 phone|             202-225-6536|
| 20|    2|               twitter|                USRepLong|
+---+-----+----------------------+-------------------------+
```

relationalize

`relationalize(root_table_name, staging_path, options, transformation_ctx="", info="", stageThreshold=0, totalThreshold=0)`

Convierte un `DynamicFrame` en un formulario que cabe en una base de datos relacional. Relacionalizar un `DynamicFrame` es especialmente útil cuando se desean mover datos de un entorno NoSQL como DynamoDB a una base de datos relacional como MySQL.

La transformación genera una lista de marcos al aplanar columnas anidadas y dinamizar las columnas de la matriz. Las columnas de matriz dinamizada se pueden unir a la tabla raíz con la clave de combinación generada durante la fase de desanidado.

- `root_table_name`: nombre de la tabla raíz.
- `staging_path`: ruta en la que el método puede almacenar las particiones de las tablas dinamizadas en formato CSV (opcional). Las tablas dinamizadas se leen desde esta ruta.
- `options`: diccionario de parámetros opcionales.
- `transformation_ctx`: cadena única que se utiliza para identificar la información del estado (opcional).
- `info`: cadena que se asociará a la notificación de errores para esta transformación (opcional).
- `stageThreshold`: cantidad de errores detectados durante esta transformación que provocarán que el proceso falle (opcional). El valor predeterminado es cero, lo que indica que el proceso no debería fallar.
- `totalThreshold`: cantidad de errores detectados hasta esta transformación incluida que provocarán que el proceso falle (opcional). El valor predeterminado es cero, lo que indica que el proceso no debería fallar.

Ejemplo: utilice relacionalizar para aplanar un esquema anidado en un **`DynamicFrame`**

Este ejemplo de código utiliza el método `relationalize` para aplanar un esquema anidado en un formulario que encaje en una base de datos relacional.

Conjunto de datos de ejemplo

El ejemplo usa un `DynamicFrame` llamado `legislators_combined` con el siguiente esquema. `legislators_combined` tiene varios campos anidados como `links`, `images` y `contact_details`, que se aplanarán con la transformación `relationalize`.

```
root
|-- role: string
|-- seats: int
|-- org_name: string
|-- links: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- url: string
|-- type: string
```

```
|-- sort_name: string
|-- area_id: string
|-- images: array
|    |-- element: struct
|    |    |-- url: string
|-- on_behalf_of_id: string
|-- other_names: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- name: string
|    |    |-- lang: string
|-- contact_details: array
|    |-- element: struct
|    |    |-- type: string
|    |    |-- value: string
|-- name: string
|-- birth_date: string
|-- organization_id: string
|-- gender: string
|-- classification: string
|-- legislative_period_id: string
|-- identifiers: array
|    |-- element: struct
|    |    |-- scheme: string
|    |    |-- identifier: string
|-- image: string
|-- given_name: string
|-- start_date: string
|-- family_name: string
|-- id: string
|-- death_date: string
|-- end_date: string
```

Código de ejemplo

```
# Example: Use relationalize to flatten
# a nested schema into a format that fits
# into a relational database.
# Replace DOC-EXAMPLE-S3-BUCKET/tmpDir with your own location.

from pyspark.context import SparkContext
from awsglue.context import GlueContext
```

```
# Create GlueContext
sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

# Apply relationalize and inspect new tables
legislators_relationalized = legislators_combined.relationalize(
    "l_root", "s3://DOC-EXAMPLE-BUCKET/tmpDir"
)
legislators_relationalized.keys()

# Compare the schema of the contact_details
# nested field to the new relationalized table that
# represents it
legislators_combined.select_fields("contact_details").printSchema()
legislators_relationalized.select("l_root_contact_details").toDF().where(
    "id = 10 or id = 75"
).orderBy(["id", "index"]).show()
```

Salida

El siguiente resultado permite comparar el esquema del campo anidado llamado `contact_details` a la tabla que creó la transformación `relationalize`. Observe que los registros de la tabla se vinculan a la tabla principal mediante una clave externa llamada `id` y una columna `index` que representa las posiciones de la matriz.

```
dict_keys(['l_root', 'l_root_images', 'l_root_links', 'l_root_other_names',
 'l_root_contact_details', 'l_root_identifiers'])

root
|-- contact_details: array
|    |-- element: struct
|    |    |-- type: string
|    |    |-- value: string

+---+-----+-----------------------+-----------------------+
| id|index|contact_details.val.type|contact_details.val.value|
+---+-----+-----------------------+-----------------------+
| 10|    0|                    fax|           202-225-4160|
| 10|    1|                  phone|           202-225-3436|
| 75|    0|                    fax|           202-225-6791|
| 75|    1|                  phone|           202-225-2861|
| 75|    2|                twitter|             RepSamFarr|
```

```
+---+-----+-----------------------+-----------------------+
```

rename_field

`rename_field(oldName, newName, transformation_ctx="", info="", stageThreshold=0, totalThreshold=0)`

Cambia el nombre de un campo en este `DynamicFrame` y devuelve un `DynamicFrame` nuevo con el campo con el nombre nuevo.

- `oldName`: la ruta completa al nodo cuyo nombre desea cambiar.

  Si el nombre antiguo tiene puntos, `RenameField` no funcionará a menos que lo ponga entre acentos graves (`). Por ejemplo, para reemplazar `this.old.name` por `thisNewName`, debería llamar a rename_field tal y como se indica a continuación:

  ```
  newDyF = oldDyF.rename_field("`this.old.name`", "thisNewName")
  ```

- `newName`: el nombre nuevo, indicado como una ruta completa.
- `transformation_ctx`: cadena única que se utiliza para identificar la información del estado (opcional).
- `info`: cadena que se asociará a la notificación de errores para esta transformación (opcional).
- `stageThreshold`: cantidad de errores detectados durante esta transformación que provocarán que el proceso falle (opcional). El valor predeterminado es cero, lo que indica que el proceso no debería fallar.
- `totalThreshold`: cantidad de errores detectados hasta esta transformación incluida que provocarán que el proceso falle (opcional). El valor predeterminado es cero, lo que indica que el proceso no debería fallar.

Ejemplo: utilice rename_field para cambiar el nombre de los campos de un **`DynamicFrame`**

Este ejemplo de código utiliza el método `rename_field` para cambiar el nombre de los campos de un `DynamicFrame`. Observe que en el ejemplo se utiliza el encadenamiento de métodos para renombrar varios campos al mismo tiempo.

> **Note**
>
> Para acceder al conjunto de datos utilizado en este ejemplo, consulte Código de ejemplo: unión de los datos y establecimiento de relaciones entre ellos y siga las instrucciones en Paso 1: Rastrear los datos del bucket de Amazon S3.

Código de ejemplo

```
# Example: Use rename_field to rename fields
# in a DynamicFrame

from pyspark.context import SparkContext
from awsglue.context import GlueContext

# Create GlueContext
sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

# Inspect the original orgs schema
orgs = glueContext.create_dynamic_frame.from_catalog(
    database="legislators", table_name="organizations_json"
)
print("Original orgs schema: ")
orgs.printSchema()

# Rename fields and view the new schema
orgs = orgs.rename_field("id", "org_id").rename_field("name", "org_name")
print("New orgs schema with renamed fields: ")
orgs.printSchema()
```

Salida

```
Original orgs schema:
root
|-- identifiers: array
|    |-- element: struct
|    |    |-- scheme: string
|    |    |-- identifier: string
|-- other_names: array
|    |-- element: struct
|    |    |-- lang: string
```

```
|    |    |-- note: string
|    |    |-- name: string
|-- id: string
|-- classification: string
|-- name: string
|-- links: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- url: string
|-- image: string
|-- seats: int
|-- type: string

New orgs schema with renamed fields:
root
|-- identifiers: array
|    |-- element: struct
|    |    |-- scheme: string
|    |    |-- identifier: string
|-- other_names: array
|    |-- element: struct
|    |    |-- lang: string
|    |    |-- note: string
|    |    |-- name: string
|-- classification: string
|-- org_id: string
|-- org_name: string
|-- links: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- url: string
|-- image: string
|-- seats: int
|-- type: string
```

resolveChoice

`resolveChoice(specs = None, choice = "" , database = None , table_name = None , transformation_ctx="", info="", stageThreshold=0, totalThreshold=0, catalog_id = None)`

Resuelve un tipo de elección dentro de este elemento `DynamicFrame` y devuelve el `DynamicFrame` nuevo.

- `specs`: lista de ambigüedades concretas que deben resolverse, cada una en forma de tupla: `(field_path, action)`.

  Hay dos formas de usar `resolveChoice`. La primera consiste en utilizar el argumento `specs` para especificar una secuencia de campos específicos y cómo resolverlos. El otro modo de `resolveChoice` es utilizar el argumento `choice` para especificar una resolución única para todos los `ChoiceTypes`.

  Valores de `specs` se especifican como tuplas compuestas de pares `(field_path, action)`. El valor de `field_path` identifica un elemento ambiguo concreto, mientras que el valor de `action` identifica la resolución correspondiente. Los posibles valores son los siguientes:

  - `cast:`*`type`*: intenta convertir todos los valores al tipo especificado. Por ejemplo: `cast:int`.
  - `make_cols`: convierte cada tipo diferenciado en una columna con el nombre *`columnName`*`_`*`type`*. Resuelve una posible ambigüedad al aplanar los datos. Por ejemplo, si `columnA` puede ser tanto un `int` como una `string`, la resolución puede consistir en producir dos columnas llamadas `columnA_int` y `columnA_string` en el elemento `DynamicFrame` resultante.
  - `make_struct`: resuelve una posible ambigüedad al utilizar una `struct` para representar los datos. Por ejemplo, si los datos de una columna pueden ser tanto un `int` como una `string`, la acción `make_struct` generará una columna de estructuras en el elemento `DynamicFrame` resultante. Cada estructura contiene tanto un `int` como una `string`.
  - `project:`*`type`*: resuelve una posible ambigüedad al proyectar todos los datos a uno de los tipos de datos posibles. Por ejemplo, si los datos de una columna pueden ser tanto un `int` como una `string`, ejecutar la acción `project:string` genera una columna en el elemento `DynamicFrame` resultante, en la que todos los valores `int` se han convertido en cadenas.

  Si en `field_path` se identifica una matriz, incluya corchetes vacíos después del nombre de la matriz para evitar ambigüedades. Por ejemplo, suponga que está trabajando con datos estructurados tal y como se indica a continuación:

```
"myList": [
  { "price": 100.00 },
  { "price": "$100.00" }
]
```

Puede seleccionar la versión numérica en vez de la de cadena del precio si configura `field_path` en `"myList[].price"` y `action` en `"cast:double"`.

> **Note**
>
> Solo se puede utilizar uno, el parámetro `specs` o el parámetro `choice`. Si el parámetro `specs` no es `None`, el parámetro `choice` tiene que ser una cadena vacía. Y viceversa, si el parámetro `choice` no es una cadena vacía, el parámetro `specs` tiene que ser `None`.

- `choice`: especifica una resolución única para todos los `ChoiceTypes`. Se puede utilizar en los casos en los que se desconozca la lista completa de `ChoiceTypes` antes del tiempo de ejecución. Además de las acciones enumeradas anteriormente para `specs`, este argumento también soporta la siguiente acción:
  - `match_catalog`: intenta convertir cada `ChoiceType` al tipo correspondiente en la tabla de Data Catalog especificada.
- `database`: la base de datos de Data Catalog que se usará con la acción `match_catalog`.
- `table_name`: la tabla de Data Catalog que se usará con la acción `match_catalog`.
- `transformation_ctx`: cadena única que se utiliza para identificar la información del estado (opcional).
- `info`: cadena que se asociará a la notificación de errores para esta transformación (opcional).
- `stageThreshold`: cantidad de errores detectados durante esta transformación que provocarán que el proceso falle (opcional). El valor predeterminado es cero, lo que indica que el proceso no debería fallar.
- `totalThreshold`: número de errores detectados hasta este proceso de transformación incluido que provocarán que el proceso se termine (opcional). Es cero de forma predeterminada, lo que indica que el proceso no debería terminarse por error.
- `catalog_id`: el ID de catálogo de Data Catalog al que se accede (el ID de cuenta de Data Catalog). Cuando se establece en `None` (valor predeterminado), utiliza el ID de catálogo de la cuenta que hace la llamada.

Ejemplo: utilice resolveChoice para gestionar una columna que contiene varios tipos

Este ejemplo de código utiliza el método `resolveChoice` para especificar cómo gestionar una columna `DynamicFrame` que contiene valores de varios tipos. El ejemplo muestra dos formas comunes de gestionar una columna con diferentes tipos:

- Convierta la columna en un solo tipo de datos.
- Conserve todos los tipos en columnas separadas.

Conjunto de datos de ejemplo

> **Note**
>
> Para acceder al conjunto de datos utilizado en este ejemplo, consulte Ejemplo de código: Preparación de datos con ResolveChoice, Lambda y ApplyMapping y siga las instrucciones en Paso 1: Rastrear los datos del bucket de Amazon S3.

El ejemplo usa un `DynamicFrame` llamado `medicare` con el siguiente esquema:

```
root
|-- drg definition: string
|-- provider id: choice
|    |-- long
|    |-- string
|-- provider name: string
|-- provider street address: string
|-- provider city: string
|-- provider state: string
|-- provider zip code: long
|-- hospital referral region description: string
|-- total discharges: long
|-- average covered charges: string
|-- average total payments: string
|-- average medicare payments: string
```

Código de ejemplo

```
# Example: Use resolveChoice to handle
# a column that contains multiple types
```

```
from pyspark.context import SparkContext
from awsglue.context import GlueContext

# Create GlueContext
sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

# Load the input data and inspect the "provider id" column
medicare = glueContext.create_dynamic_frame.from_catalog(
    database="payments", table_name="medicare_hospital_provider_csv"
)
print("Inspect the provider id column:")
medicare.toDF().select("provider id").show()

# Cast provider id to type long
medicare_resolved_long = medicare.resolveChoice(specs=[("provider id", "cast:long")])
print("Schema after casting provider id to type long:")
medicare_resolved_long.printSchema()
medicare_resolved_long.toDF().select("provider id").show()

# Create separate columns
# for each provider id type
medicare_resolved_cols = medicare.resolveChoice(choice="make_cols")
print("Schema after creating separate columns for each type:")
medicare_resolved_cols.printSchema()
medicare_resolved_cols.toDF().select("provider id_long", "provider id_string").show()
```

Salida

```
Inspect the 'provider id' column:
+-----------+
|provider id|
+-----------+
|   [10001,]|
|   [10005,]|
|   [10006,]|
|   [10011,]|
|   [10016,]|
|   [10023,]|
|   [10029,]|
|   [10033,]|
|   [10039,]|
```

```
|   [10040,]|
|   [10046,]|
|   [10055,]|
|   [10056,]|
|   [10078,]|
|   [10083,]|
|   [10085,]|
|   [10090,]|
|   [10092,]|
|   [10100,]|
|   [10103,]|
+-----------+
only showing top 20 rows

Schema after casting 'provider id' to type long:
root
|-- drg definition: string
|-- provider id: long
|-- provider name: string
|-- provider street address: string
|-- provider city: string
|-- provider state: string
|-- provider zip code: long
|-- hospital referral region description: string
|-- total discharges: long
|-- average covered charges: string
|-- average total payments: string
|-- average medicare payments: string

+-----------+
|provider id|
+-----------+
|      10001|
|      10005|
|      10006|
|      10011|
|      10016|
|      10023|
|      10029|
|      10033|
|      10039|
|      10040|
|      10046|
|      10055|
```

```
|      10056|
|      10078|
|      10083|
|      10085|
|      10090|
|      10092|
|      10100|
|      10103|
+-----------+
only showing top 20 rows

Schema after creating separate columns for each type:
root
|-- drg definition: string
|-- provider id_string: string
|-- provider id_long: long
|-- provider name: string
|-- provider street address: string
|-- provider city: string
|-- provider state: string
|-- provider zip code: long
|-- hospital referral region description: string
|-- total discharges: long
|-- average covered charges: string
|-- average total payments: string
|-- average medicare payments: string

+----------------+------------------+
|provider id_long|provider id_string|
+----------------+------------------+
|           10001|              null|
|           10005|              null|
|           10006|              null|
|           10011|              null|
|           10016|              null|
|           10023|              null|
|           10029|              null|
|           10033|              null|
|           10039|              null|
|           10040|              null|
|           10046|              null|
|           10055|              null|
|           10056|              null|
|           10078|              null|
```

```
|           10083|              null|
|           10085|              null|
|           10090|              null|
|           10092|              null|
|           10100|              null|
|           10103|              null|
+----------------+------------------+
only showing top 20 rows
```

select_fields

`select_fields(paths, transformation_ctx="", info="", stageThreshold=0, totalThreshold=0)`

Devuelve un `DynamicFrame` nuevo que contiene los campos seleccionados.

- `paths`: una lista de cadenas. Cada cadena es una ruta a un nodo de nivel superior que desea seleccionar.
- `transformation_ctx`: cadena única que se utiliza para identificar la información del estado (opcional).
- `info`: cadena que se asociará a la notificación de errores para esta transformación (opcional).
- `stageThreshold`: cantidad de errores detectados durante esta transformación que provocarán que el proceso falle (opcional). El valor predeterminado es cero, lo que indica que el proceso no debería fallar.
- `totalThreshold`: cantidad de errores detectados hasta esta transformación incluida que provocarán que el proceso falle (opcional). El valor predeterminado es cero, lo que indica que el proceso no debería fallar.

Ejemplo: uso de select_fields para crear un **`DynamicFrame`** nuevo con los campos elegidos

En el siguiente ejemplo de código, se muestra cómo utilizar el método `select_fields` para crear un `DynamicFrame` nuevo con una lista seleccionada de campos de un `DynamicFrame` existente.

> **Note**
>
> Para acceder al conjunto de datos utilizado en este ejemplo, consulte Código de ejemplo: unión de los datos y establecimiento de relaciones entre ellos y siga las instrucciones en Paso 1: Rastrear los datos del bucket de Amazon S3.

```
# Example: Use select_fields to select specific fields from a DynamicFrame

from pyspark.context import SparkContext
from awsglue.context import GlueContext

# Create GlueContext
sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

# Create a DynamicFrame and view its schema
persons = glueContext.create_dynamic_frame.from_catalog(
    database="legislators", table_name="persons_json"
)
print("Schema for the persons DynamicFrame:")
persons.printSchema()

# Create a new DynamicFrame with chosen fields
names = persons.select_fields(paths=["family_name", "given_name"])
print("Schema for the names DynamicFrame, created with select_fields:")
names.printSchema()
names.toDF().show()
```

Salida

```
Schema for the persons DynamicFrame:
root
|-- family_name: string
|-- name: string
|-- links: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- url: string
|-- gender: string
|-- image: string
|-- identifiers: array
```

```
|    |-- element: struct
|    |    |-- scheme: string
|    |    |-- identifier: string
|-- other_names: array
|    |-- element: struct
|    |    |-- lang: string
|    |    |-- note: string
|    |    |-- name: string
|-- sort_name: string
|-- images: array
|    |-- element: struct
|    |    |-- url: string
|-- given_name: string
|-- birth_date: string
|-- id: string
|-- contact_details: array
|    |-- element: struct
|    |    |-- type: string
|    |    |-- value: string
|-- death_date: string

Schema for the names DynamicFrame:
root
|-- family_name: string
|-- given_name: string

+-----------+----------+
|family_name|given_name|
+-----------+----------+
|    Collins|   Michael|
|   Huizenga|      Bill|
|    Clawson|    Curtis|
|    Solomon|    Gerald|
|     Rigell|    Edward|
|      Crapo|   Michael|
|      Hutto|      Earl|
|      Ertel|     Allen|
|     Minish|    Joseph|
|    Andrews|    Robert|
|     Walden|      Greg|
|      Kazen|   Abraham|
|     Turner|   Michael|
|      Kolbe|     James|
|  Lowenthal|      Alan|
```

```
|    Capuano|   Michael|
|   Schrader|      Kurt|
|     Nadler|   Jerrold|
|     Graves|       Tom|
|   McMillan|      John|
+-----------+----------+
only showing top 20 rows
```

simplify_ddb_json

`simplify_ddb_json(): DynamicFrame`

Simplifica las columnas anidadas en un `DynamicFrame` que se encuentran de manera específica en la estructura de JSON de DynamoDB y devuelve un nuevo `DynamicFrame` simplificado. Si hay varios tipos y tipos de mapas una lista de tipos, no se simplificarán los elementos de la lista. Tenga en cuenta que este es un tipo específico de transformación que se comporta de manera distinta a la transformación `unnest` regular y que necesita que los datos ya se encuentren en la estructura JSON de DynamoDB. Para obtener más información, consulte JSON de DynamoDB.

Por ejemplo, el esquema de una lectura de una exportación con la estructura JSON de DynamoDB puede parecerse al siguiente:

```
root
|-- Item: struct
|    |-- parentMap: struct
|    |    |-- M: struct
|    |    |    |-- childMap: struct
|    |    |    |    |-- M: struct
|    |    |    |    |    |-- appName: struct
|    |    |    |    |    |    |-- S: string
|    |    |    |    |    |-- packageName: struct
|    |    |    |    |    |    |-- S: string
|    |    |    |    |    |-- updatedAt: struct
|    |    |    |    |    |    |-- N: string
|    |-- strings: struct
|    |    |-- SS: array
|    |    |    |-- element: string
|    |-- numbers: struct
|    |    |-- NS: array
|    |    |    |-- element: string
|    |-- binaries: struct
```

```
|    |    |-- BS: array
|    |    |    |-- element: string
|    |-- isDDBJson: struct
|    |    |-- BOOL: boolean
|    |-- nullValue: struct
|    |    |-- NULL: boolean
```

La transformación `simplify_ddb_json()` convertiría esto en:

```
root
|-- parentMap: struct
|    |-- childMap: struct
|    |    |-- appName: string
|    |    |-- packageName: string
|    |    |-- updatedAt: string
|-- strings: array
|    |-- element: string
|-- numbers: array
|    |-- element: string
|-- binaries: array
|    |-- element: string
|-- isDDBJson: boolean
|-- nullValue: null
```

Por ejemplo: utilice simplify_ddb_json para invocar una simplificación JSON de DynamoDB.

Este ejemplo de código utiliza el método `simplify_ddb_json` para utilizar el conecto de exportación de DynamoDB de AWS Glue, invocar una simplificación JSON de DynamoDB e imprimir el número de pariciones.

Código de ejemplo

```
from pyspark.context import SparkContext
from awsglue.context import GlueContext

sc = SparkContext()
glueContext = GlueContext(sc)

dynamicFrame = glueContext.create_dynamic_frame.from_options(
    connection_type = "dynamodb",
    connection_options = {
        'dynamodb.export': 'ddb',
```

```
        'dynamodb.tableArn': '<table arn>',
        'dynamodb.s3.bucket': '<bucket name>',
        'dynamodb.s3.prefix': '<bucket prefix>',
        'dynamodb.s3.bucketOwner': '<account_id of bucket>'
    }
)
simplified = dynamicFrame.simplify_ddb_json()
print(simplified.getNumPartitions())
```

spigot

`spigot(path, options={})`

Escribe registros de ejemplo en un destino específico para ayudarlo a verificar las transformaciones realizadas por su trabajo.

- `path`: ruta del destino en el que se escribirá (obligatorio).
- `options`: pares valor-clave que especifican opciones (opcional). La opción `"topk"` especifica que deben escribirse los primeros registros k. La opción `"prob"` especifica la probabilidad de que se elija un registro determinado (como decimal). Puede usarlo para seleccionar registros para escribir.
- `transformation_ctx`: cadena única que se utiliza para identificar la información del estado (opcional).

Ejemplo: utilice spigot para escribir campos de ejemplo desde un **`DynamicFrame`** en Amazon S3

Este ejemplo de código utiliza el método `spigot` para escribir registros de ejemplo en un bucket de Amazon S3 después de aplicar la transformación `select_fields`.

Conjunto de datos de ejemplo

> **Note**
>
> Para acceder al conjunto de datos utilizado en este ejemplo, consulte Código de ejemplo: unión de los datos y establecimiento de relaciones entre ellos y siga las instrucciones en Paso 1: Rastrear los datos del bucket de Amazon S3.

El ejemplo usa un `DynamicFrame` llamado `persons` con el siguiente esquema:

```
root
```

```
|-- family_name: string
|-- name: string
|-- links: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- url: string
|-- gender: string
|-- image: string
|-- identifiers: array
|    |-- element: struct
|    |    |-- scheme: string
|    |    |-- identifier: string
|-- other_names: array
|    |-- element: struct
|    |    |-- lang: string
|    |    |-- note: string
|    |    |-- name: string
|-- sort_name: string
|-- images: array
|    |-- element: struct
|    |    |-- url: string
|-- given_name: string
|-- birth_date: string
|-- id: string
|-- contact_details: array
|    |-- element: struct
|    |    |-- type: string
|    |    |-- value: string
|-- death_date: string
```

Código de ejemplo

```
# Example: Use spigot to write sample records
# to a destination during a transformation
# from pyspark.context import SparkContext.
# Replace DOC-EXAMPLE-BUCKET with your own location.

from pyspark.context import SparkContext
from awsglue.context import GlueContext

# Create GlueContext
sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)
```

```
# Load table data into a DynamicFrame
persons = glueContext.create_dynamic_frame.from_catalog(
    database="legislators", table_name="persons_json"
)

# Perform the select_fields on the DynamicFrame
persons = persons.select_fields(paths=["family_name", "given_name", "birth_date"])

# Use spigot to write a sample of the transformed data
# (the first 10 records)
spigot_output = persons.spigot(
    path="s3://DOC-EXAMPLE-BUCKET", options={"topk": 10}
)
```

Salida

A continuación, se muestra un ejemplo de los datos que `spigot` escribe en Amazon S3. Como se especifica el código de ejemplo `options={"topk": 10}`, los datos de la muestra contienen los 10 primeros registros.

```
{"family_name":"Collins","given_name":"Michael","birth_date":"1944-10-15"}
{"family_name":"Huizenga","given_name":"Bill","birth_date":"1969-01-31"}
{"family_name":"Clawson","given_name":"Curtis","birth_date":"1959-09-28"}
{"family_name":"Solomon","given_name":"Gerald","birth_date":"1930-08-14"}
{"family_name":"Rigell","given_name":"Edward","birth_date":"1960-05-28"}
{"family_name":"Crapo","given_name":"Michael","birth_date":"1951-05-20"}
{"family_name":"Hutto","given_name":"Earl","birth_date":"1926-05-12"}
{"family_name":"Ertel","given_name":"Allen","birth_date":"1937-11-07"}
{"family_name":"Minish","given_name":"Joseph","birth_date":"1916-09-01"}
{"family_name":"Andrews","given_name":"Robert","birth_date":"1957-08-04"}
```

split_fields

`split_fields(paths, name1, name2, transformation_ctx="", info="", stageThreshold=0, totalThreshold=0)`

Devuelve una nueva `DynamicFrameCollection` que contiene dos `DynamicFrames`. El primer `DynamicFrame` contiene todos los nodos que se han dividido y el segundo contiene los nodos que quedan.

- `paths`: lista de cadenas, cada una de las cuales es una ruta completa a un nodo que se desea dividir en un elemento `DynamicFrame` nuevo.
- `name1`: cadena de nombre para el `DynamicFrame` que se divide.
- `name2`: cadena de nombre para el elemento `DynamicFrame` que queda después de que se hayan dividido los nodos especificados.
- `transformation_ctx`: cadena única que se utiliza para identificar la información del estado (opcional).
- `info`: cadena que se asociará a la notificación de errores para esta transformación (opcional).
- `stageThreshold`: cantidad de errores detectados durante esta transformación que provocarán que el proceso falle (opcional). El valor predeterminado es cero, lo que indica que el proceso no debería fallar.
- `totalThreshold`: cantidad de errores detectados hasta esta transformación incluida que provocarán que el proceso falle (opcional). El valor predeterminado es cero, lo que indica que el proceso no debería fallar.

Ejemplo: utilice split_fields para dividir los campos seleccionados en un campo **`DynamicFrame`** independiente

Este ejemplo de código usa el método `split_fields` para dividir una lista de campos especificados en una lista `DynamicFrame` independiente.

Conjunto de datos de ejemplo

El ejemplo usa un `DynamicFrame` llamado `l_root_contact_details` que proviene de una colección denominada `legislators_relationalized`.

`l_root_contact_details` tiene el siguiente esquema y entradas.

```
root
|-- id: long
|-- index: int
|-- contact_details.val.type: string
|-- contact_details.val.value: string

+---+-----+------------------------+-------------------------+
| id|index|contact_details.val.type|contact_details.val.value|
+---+-----+------------------------+-------------------------+
```

```
|  1|    0|                  phone|             202-225-5265|
|  1|    1|                twitter|              kathyhochul|
|  2|    0|                  phone|             202-225-3252|
|  2|    1|                twitter|            repjackyrosen|
|  3|    0|                    fax|             202-225-1314|
|  3|    1|                  phone|             202-225-3772|
...
```

Código de ejemplo

```
# Example: Use split_fields to split selected
# fields into a separate DynamicFrame

from pyspark.context import SparkContext
from awsglue.context import GlueContext

# Create GlueContext
sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

# Load the input DynamicFrame and inspect its schema
frame_to_split = legislators_relationalized.select("l_root_contact_details")
print("Inspect the input DynamicFrame schema:")
frame_to_split.printSchema()

# Split id and index fields into a separate DynamicFrame
split_fields_collection = frame_to_split.split_fields(["id", "index"], "left", "right")

# Inspect the resulting DynamicFrames
print("Inspect the schemas of the DynamicFrames created with split_fields:")
split_fields_collection.select("left").printSchema()
split_fields_collection.select("right").printSchema()
```

Salida

```
Inspect the input DynamicFrame's schema:
root
|-- id: long
|-- index: int
|-- contact_details.val.type: string
|-- contact_details.val.value: string

Inspect the schemas of the DynamicFrames created with split_fields:
```

```
root
|-- id: long
|-- index: int

root
|-- contact_details.val.type: string
|-- contact_details.val.value: string
```

split_rows

`split_rows(comparison_dict, name1, name2, transformation_ctx="", info="", stageThreshold=0, totalThreshold=0)`

Divide una o más filas de un `DynamicFrame` en un nuevo `DynamicFrame`.

El método devuelve un nuevo `DynamicFrameCollection` que contiene dos `DynamicFrames`. El primer `DynamicFrame` contiene todas las filas que se han dividido y el segundo contiene las filas que quedan.

- `comparison_dict`: diccionario donde la clave es una ruta a una columna y el valor es otro diccionario que mapea comparadores con valores con los que se compara el valor de columna. Por ejemplo, `{"age": {">": 10, "<": 20}}` divide todas las filas cuyo valor en la columna antigüedad sea superior a 10 e inferior a 20.
- `name1`: cadena de nombre para el `DynamicFrame` que se divide.
- `name2`: cadena de nombre para el elemento `DynamicFrame` que queda después de que se hayan dividido los nodos especificados.
- `transformation_ctx`: cadena única que se utiliza para identificar la información del estado (opcional).
- `info`: cadena que se asociará a la notificación de errores para esta transformación (opcional).
- `stageThreshold`: cantidad de errores detectados durante esta transformación que provocarán que el proceso falle (opcional). El valor predeterminado es cero, lo que indica que el proceso no debería fallar.
- `totalThreshold`: cantidad de errores detectados hasta esta transformación incluida que provocarán que el proceso falle (opcional). El valor predeterminado es cero, lo que indica que el proceso no debería fallar.

Ejemplo: utilice split_rows para dividir filas en un **`DynamicFrame`**

En este ejemplo de código se utiliza el método `split_rows` para dividir las filas en un `DynamicFrame` en función del valor del campo `id`.

Conjunto de datos de ejemplo

El ejemplo usa un `DynamicFrame` llamado `l_root_contact_details` que se selecciona de una colección denominada `legislators_relationalized`.

`l_root_contact_details` tiene el siguiente esquema y entradas.

```
root
|-- id: long
|-- index: int
|-- contact_details.val.type: string
|-- contact_details.val.value: string

+---+-----+-----------------------+-----------------------+
| id|index|contact_details.val.type|contact_details.val.value|
+---+-----+-----------------------+-----------------------+
|  1|    0|                  phone|             202-225-5265|
|  1|    1|                twitter|              kathyhochul|
|  2|    0|                  phone|             202-225-3252|
|  2|    1|                twitter|            repjackyrosen|
|  3|    0|                    fax|             202-225-1314|
|  3|    1|                  phone|             202-225-3772|
|  3|    2|                twitter|          MikeRossUpdates|
|  4|    0|                    fax|             202-225-1314|
|  4|    1|                  phone|             202-225-3772|
|  4|    2|                twitter|          MikeRossUpdates|
|  5|    0|                    fax|             202-225-1314|
|  5|    1|                  phone|             202-225-3772|
|  5|    2|                twitter|          MikeRossUpdates|
|  6|    0|                    fax|             202-225-1314|
|  6|    1|                  phone|             202-225-3772|
|  6|    2|                twitter|          MikeRossUpdates|
|  7|    0|                    fax|             202-225-1314|
|  7|    1|                  phone|             202-225-3772|
|  7|    2|                twitter|          MikeRossUpdates|
|  8|    0|                    fax|             202-225-1314|
+---+-----+-----------------------+-----------------------+
```

Código de ejemplo

```
# Example: Use split_rows to split up
# rows in a DynamicFrame based on value

from pyspark.context import SparkContext
from awsglue.context import GlueContext

# Create GlueContext
sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

# Retrieve the DynamicFrame to split
frame_to_split = legislators_relationalized.select("l_root_contact_details")

# Split up rows by ID
split_rows_collection = frame_to_split.split_rows({"id": {">": 10}}, "high", "low")

# Inspect the resulting DynamicFrames
print("Inspect the DynamicFrame that contains IDs < 10")
split_rows_collection.select("low").toDF().show()
print("Inspect the DynamicFrame that contains IDs > 10")
split_rows_collection.select("high").toDF().show()
```

Salida

```
Inspect the DynamicFrame that contains IDs < 10
+---+-----+-----------------------+-----------------------+
| id|index|contact_details.val.type|contact_details.val.value|
+---+-----+-----------------------+-----------------------+
|  1|    0|                  phone|             202-225-5265|
|  1|    1|                twitter|              kathyhochul|
|  2|    0|                  phone|             202-225-3252|
|  2|    1|                twitter|            repjackyrosen|
|  3|    0|                    fax|             202-225-1314|
|  3|    1|                  phone|             202-225-3772|
|  3|    2|                twitter|          MikeRossUpdates|
|  4|    0|                    fax|             202-225-1314|
|  4|    1|                  phone|             202-225-3772|
|  4|    2|                twitter|          MikeRossUpdates|
|  5|    0|                    fax|             202-225-1314|
|  5|    1|                  phone|             202-225-3772|
|  5|    2|                twitter|          MikeRossUpdates|
```

```
|  6|    0|                     fax|             202-225-1314|
|  6|    1|                   phone|             202-225-3772|
|  6|    2|                 twitter|          MikeRossUpdates|
|  7|    0|                     fax|             202-225-1314|
|  7|    1|                   phone|             202-225-3772|
|  7|    2|                 twitter|          MikeRossUpdates|
|  8|    0|                     fax|             202-225-1314|
+---+-----+------------------------+-------------------------+
only showing top 20 rows

Inspect the DynamicFrame that contains IDs > 10
+---+-----+------------------------+-------------------------+
| id|index|contact_details.val.type|contact_details.val.value|
+---+-----+------------------------+-------------------------+
| 11|    0|                   phone|             202-225-5476|
| 11|    1|                 twitter|            RepDavidYoung|
| 12|    0|                   phone|             202-225-4035|
| 12|    1|                 twitter|           RepStephMurphy|
| 13|    0|                     fax|             202-226-0774|
| 13|    1|                   phone|             202-225-6335|
| 14|    0|                     fax|             202-226-0774|
| 14|    1|                   phone|             202-225-6335|
| 15|    0|                     fax|             202-226-0774|
| 15|    1|                   phone|             202-225-6335|
| 16|    0|                     fax|             202-226-0774|
| 16|    1|                   phone|             202-225-6335|
| 17|    0|                     fax|             202-226-0774|
| 17|    1|                   phone|             202-225-6335|
| 18|    0|                     fax|             202-226-0774|
| 18|    1|                   phone|             202-225-6335|
| 19|    0|                     fax|             202-226-0774|
| 19|    1|                   phone|             202-225-6335|
| 20|    0|                     fax|             202-226-0774|
| 20|    1|                   phone|             202-225-6335|
+---+-----+------------------------+-------------------------+
only showing top 20 rows
```

unbox

`unbox(path, format, transformation_ctx="", info="", stageThreshold=0, totalThreshold=0, **options)`

Realiza la conversión unboxing (vuelve a formatear) de un campo de cadena en un elemento `DynamicFrame` y devuelve un elemento `DynamicFrame` nuevo que contiene el elemento `DynamicRecords` de conversión unboxing.

Un `DynamicRecord` representa un registro lógico en un `DynamicFrame`. Es similar a una fila en un `DataFrame` de Apache Spark, salvo que es autodescriptivo y se puede utilizar para datos que no se ajustan a un esquema fijo.

- `path`: ruta completa al nodo de cadena a cuyo nombre se desea aplicar la conversión unboxing.
- `format`: una especificación de formato (opcional). La utiliza con una conexión de Amazon S3 o AWS Glue que admite diversos formatos. Consulte en Opciones de formato de datos para las entradas y las salidas en AWS Glue para Spark los formatos que se admiten.
- `transformation_ctx`: cadena única que se utiliza para identificar la información del estado (opcional).
- `info`: cadena que se asociará a la notificación de errores para esta transformación (opcional).
- `stageThreshold`: cantidad de errores detectados durante esta transformación que provocarán que el proceso falle (opcional). El valor predeterminado es cero, lo que indica que el proceso no debería fallar.
- `totalThreshold`: cantidad de errores detectados hasta esta transformación incluida que provocarán que el proceso falle (opcional). El valor predeterminado es cero, lo que indica que el proceso no debería fallar.
- `options`: una o varias de las siguientes:
  - `separator`: cadena que contiene el carácter separador.
  - `escaper`: cadena que contiene el carácter de escape.
  - `skipFirst`: valor booleano que indica si debe omitirse la primera instancia.
  - `withSchema`: una cadena que contiene una representación JSON del esquema del nodo. El formato de la representación JSON de un esquema se define mediante el resultado de `StructType.json()`.
  - `withHeader`: valor booleano que indica si se incluye un encabezado.

Ejemplo: utilice la conversión unboxing para convertir un campo de cadena en una estructura

Este ejemplo de código utiliza el método `unbox` para la conversión unboxing o volver a formatear un campo de cadena en un `DynamicFrame` en un campo de tipo estructura.

Conjunto de datos de ejemplo

El ejemplo usa un `DynamicFrame` llamado `mapped_with_string` con el siguiente esquema y entradas.

Observe el campo denominado `AddressString`. Este es el campo que el ejemplo convierte en una estructura.

```
root
|-- Average Total Payments: string
|-- AddressString: string
|-- Average Covered Charges: string
|-- DRG Definition: string
|-- Average Medicare Payments: string
|-- Hospital Referral Region Description: string
|-- Address: struct
|    |-- Zip.Code: string
|    |-- City: string
|    |-- Array: array
|    |    |-- element: string
|    |-- State: string
|    |-- Street: string
|-- Provider Id: string
|-- Total Discharges: string
|-- Provider Name: string

+----------------------+--------------------+-----------------------
+--------------------+------------------------+-----------------------------------
+--------------------+-----------+----------------+--------------------+
|Average Total Payments|       AddressString|Average Covered Charges|      DRG
 Definition|Average Medicare Payments|Hospital Referral Region Description|
 Address|Provider Id|Total Discharges|       Provider Name|
+----------------------+--------------------+-----------------------
+--------------------+------------------------+-----------------------------------
+--------------------+-----------+----------------+--------------------+
|              $5777.24|{"Street": "1108 ...|              $32963.07|039 -
 EXTRACRANIA...|                $4763.73|                        AL - Dothan|[36301,
 DOTHAN, [...|      10001|              91|SOUTHEAST ALABAMA...|
```

```
|              $5787.57|{"Street": "2505 ...|              $15131.85|039 -
 EXTRACRANIA...|                 $4976.71|                    AL - Birmingham|[35957,
 BOAZ, [25...|      10005|              14|MARSHALL MEDICAL ...|
|              $5434.95|{"Street": "205 M...|              $37560.37|039 -
 EXTRACRANIA...|                 $4453.79|                    AL - Birmingham|[35631,
 FLORENCE,...|      10006|              24|ELIZA COFFEE MEMO...|
|              $5417.56|{"Street": "50 ME...|              $13998.28|039 -
 EXTRACRANIA...|                 $4129.16|                    AL - Birmingham|[35235,
 BIRMINGHA...|      10011|              25|   ST VINCENT'S EAST|
...
```

Código de ejemplo

```
# Example: Use unbox to unbox a string field
# into a struct in a DynamicFrame

from pyspark.context import SparkContext
from awsglue.context import GlueContext

# Create GlueContext
sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

unboxed = mapped_with_string.unbox("AddressString", "json")
unboxed.printSchema()
unboxed.toDF().show()
```

Salida

```
root
|-- Average Total Payments: string
|-- AddressString: struct
|    |-- Street: string
|    |-- City: string
|    |-- State: string
|    |-- Zip.Code: string
|    |-- Array: array
|    |    |-- element: string
|-- Average Covered Charges: string
|-- DRG Definition: string
|-- Average Medicare Payments: string
|-- Hospital Referral Region Description: string
```

```
|-- Address: struct
|    |-- Zip.Code: string
|    |-- City: string
|    |-- Array: array
|    |    |-- element: string
|    |-- State: string
|    |-- Street: string
|-- Provider Id: string
|-- Total Discharges: string
|-- Provider Name: string

+----------------------+--------------------+-----------------------
+--------------------+------------------------+-----------------------------------
+--------------------+-----------+---------------+--------------------+
|Average Total Payments|       AddressString|Average Covered Charges|      DRG
 Definition|Average Medicare Payments|Hospital Referral Region Description|
 Address|Provider Id|Total Discharges|       Provider Name|
+----------------------+--------------------+-----------------------
+--------------------+------------------------+-----------------------------------
+--------------------+-----------+---------------+--------------------+
|              $5777.24|[1108 ROSS CLARK ...|              $32963.07|039 -
 EXTRACRANIA...|                $4763.73|                        AL - Dothan|[36301,
 DOTHAN, [...|      10001|             91|SOUTHEAST ALABAMA...|
|              $5787.57|[2505 U S HIGHWAY...|              $15131.85|039 -
 EXTRACRANIA...|                $4976.71|                    AL - Birmingham|[35957,
 BOAZ, [25...|      10005|             14|MARSHALL MEDICAL ...|
|              $5434.95|[205 MARENGO STRE...|              $37560.37|039 -
 EXTRACRANIA...|                $4453.79|                    AL - Birmingham|[35631,
 FLORENCE,...|      10006|             24|ELIZA COFFEE MEMO...|
|              $5417.56|[50 MEDICAL PARK ...|              $13998.28|039 -
 EXTRACRANIA...|                $4129.16|                    AL - Birmingham|[35235,
 BIRMINGHA...|      10011|             25|   ST VINCENT'S EAST|
|              $5658.33|[1000 FIRST STREE...|              $31633.27|039 -
 EXTRACRANIA...|                $4851.44|                    AL - Birmingham|[35007,
 ALABASTER...|      10016|             18|SHELBY BAPTIST ME...|
|              $6653.80|[2105 EAST SOUTH ...|              $16920.79|039 -
 EXTRACRANIA...|                $5374.14|                    AL - Montgomery|[36116,
 MONTGOMER...|      10023|             67|BAPTIST MEDICAL C...|
|              $5834.74|[2000 PEPPERELL P...|              $11977.13|039 -
 EXTRACRANIA...|                $4761.41|                    AL - Birmingham|[36801,
 OPELIKA, ...|      10029|             51|EAST ALABAMA MEDI...|
|              $8031.12|[619 SOUTH 19TH S...|              $35841.09|039 -
 EXTRACRANIA...|                $5858.50|                    AL - Birmingham|[35233,
 BIRMINGHA...|      10033|             32|UNIVERSITY OF ALA...|
```

```
|              $6113.38|[101 SIVLEY RD, H...|              $28523.39|039 -
 EXTRACRANIA...|                $5228.40|                     AL - Huntsville|[35801,
 HUNTSVILL...|      10039|             135| HUNTSVILLE HOSPITAL|
|              $5541.05|[1007 GOODYEAR AV...|              $75233.38|039 -
 EXTRACRANIA...|                $4386.94|                     AL - Birmingham|[35903,
 GADSDEN, ...|      10040|              34|GADSDEN REGIONAL ...|
|              $5461.57|[600 SOUTH THIRD ...|              $67327.92|039 -
 EXTRACRANIA...|                $4493.57|                     AL - Birmingham|[35901,
 GADSDEN, ...|      10046|              14|RIVERVIEW REGIONA...|
|              $5356.28|[4370 WEST MAIN S...|              $39607.28|039 -
 EXTRACRANIA...|                $4408.20|                         AL - Dothan|[36305,
 DOTHAN, [...|      10055|              45|    FLOWERS HOSPITAL|
|              $5374.65|[810 ST VINCENT'S...|              $22862.23|039 -
 EXTRACRANIA...|                $4186.02|                     AL - Birmingham|[35205,
 BIRMINGHA...|      10056|              43|ST VINCENT'S BIRM...|
|              $5366.23|[400 EAST 10TH ST...|              $31110.85|039 -
 EXTRACRANIA...|                $4376.23|                     AL - Birmingham|[36207,
 ANNISTON,...|      10078|              21|NORTHEAST ALABAMA...|
|              $5282.93|[1613 NORTH MCKEN...|              $25411.33|039 -
 EXTRACRANIA...|                $4383.73|                         AL - Mobile|[36535,
 FOLEY, [1...|      10083|              15|SOUTH BALDWIN REG...|
|              $5676.55|[1201 7TH STREET ...|               $9234.51|039 -
 EXTRACRANIA...|                $4509.11|                     AL - Huntsville|[35609,
 DECATUR, ...|      10085|              27|DECATUR GENERAL H...|
|              $5930.11|[6801 AIRPORT BOU...|              $15895.85|039 -
 EXTRACRANIA...|                $3972.85|                         AL - Mobile|[36608,
 MOBILE, [...|      10090|              27| PROVIDENCE HOSPITAL|
|              $6192.54|[809 UNIVERSITY B...|              $19721.16|039 -
 EXTRACRANIA...|                $5179.38|                     AL - Tuscaloosa|[35401,
 TUSCALOOS...|      10092|              31|D C H REGIONAL ME...|
|              $4968.00|[750 MORPHY AVENU...|              $10710.88|039 -
 EXTRACRANIA...|                $3898.88|                         AL - Mobile|[36532,
 FAIRHOPE,...|      10100|              18|     THOMAS HOSPITAL|
|              $5996.00|[701 PRINCETON AV...|              $51343.75|039 -
 EXTRACRANIA...|                $4962.45|                     AL - Birmingham|[35211,
 BIRMINGHA...|      10103|              33|BAPTIST MEDICAL C...|
+----------------------+--------------------+-----------------------
+--------------------+------------------------+-----------------------------------
+--------------------+----------+---------------+--------------------+
only showing top 20 rows
```

unión

`union(frame1, frame2, transformation_ctx = "", info = "", stageThreshold = 0, totalThreshold = 0)`

Unir dos DynamicFrames. Devuelve un DynamicFrame que contiene todos los registros de ambos DynamicFrames de entrada. Esta transformación puede devolver resultados diferentes de la unión de dos DataFrames con datos equivalentes. Si necesita el comportamiento de unión del DataFrame de Spark, considere usar `toDF`.

- `frame1`: primer DynamicFrame en unirse.
- `frame2`: segundo DynamicFrame en unirse.
- `transformation_ctx`: (opcional) una cadena única que se utiliza para identificar información de estadísticas/estado
- `info`: (opcional) una cadena que está asociada a errores en la transformación
- `stageThreshold`: (opcional) un número máximo de errores en la transformación hasta que se produzca un error en el procesamiento
- `totalThreshold`: (opcional) un número máximo de errores en total hasta que se produzca un error en el procesamiento.

unnest

`unnest(transformation_ctx="", info="", stageThreshold=0, totalThreshold=0)`

Desanida los objetos anidados de un elemento `DynamicFrame`, que los convierte en objetos de nivel superior y devuelve un elemento `DynamicFrame` desanidado nuevo.

- `transformation_ctx`: cadena única que se utiliza para identificar la información del estado (opcional).
- `info`: cadena que se asociará a la notificación de errores para esta transformación (opcional).
- `stageThreshold`: cantidad de errores detectados durante esta transformación que provocarán que el proceso falle (opcional). El valor predeterminado es cero, lo que indica que el proceso no debería fallar.
- `totalThreshold`: cantidad de errores detectados hasta esta transformación incluida que provocarán que el proceso falle (opcional). El valor predeterminado es cero, lo que indica que el proceso no debería fallar.

Ejemplo: utilice desanidar para convertir los campos anidados en campos de nivel superior

Este ejemplo de código utiliza el método `unnest` para desanidar todos los campos anidados en un `DynamicFrame` en campos de nivel superior.

Conjunto de datos de ejemplo

El ejemplo usa un `DynamicFrame` llamado `mapped_medicare` con el siguiente esquema. Observe que el campo `Address` es el único campo que contiene datos anidados.

```
root
|-- Average Total Payments: string
|-- Average Covered Charges: string
|-- DRG Definition: string
|-- Average Medicare Payments: string
|-- Hospital Referral Region Description: string
|-- Address: struct
|    |-- Zip.Code: string
|    |-- City: string
|    |-- Array: array
|    |    |-- element: string
|    |-- State: string
|    |-- Street: string
|-- Provider Id: string
|-- Total Discharges: string
|-- Provider Name: string
```

Código de ejemplo

```
# Example: Use unnest to unnest nested
# objects in a DynamicFrame

from pyspark.context import SparkContext
from awsglue.context import GlueContext

# Create GlueContext
sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

# Unnest all nested fields
unnested = mapped_medicare.unnest()
unnested.printSchema()
```

Salida

```
root
|-- Average Total Payments: string
|-- Average Covered Charges: string
|-- DRG Definition: string
|-- Average Medicare Payments: string
|-- Hospital Referral Region Description: string
|-- Address.Zip.Code: string
|-- Address.City: string
|-- Address.Array: array
|    |-- element: string
|-- Address.State: string
|-- Address.Street: string
|-- Provider Id: string
|-- Total Discharges: string
|-- Provider Name: string
```

unnest_ddb_json

Desanida las columnas anidadas de un elemento `DynamicFrame` que se encuentren específicamente en la estructura JSON de DynamoDB y devuelve un nuevo elemento `DynamicFrame` no anidado. Las columnas que sean de una matriz de tipos de estructuras no se desanidarán. Tenga en cuenta que esta transformación de desanidamiento es un tipo específico que se comporta de modo diferente a la transformación `unnest` normal y requiere que los datos ya estén en la estructura JSON de DynamoDB. Para obtener más información, consulte JSON de DynamoDB.

`unnest_ddb_json(transformation_ctx="", info="", stageThreshold=0, totalThreshold=0)`

- `transformation_ctx`: cadena única que se utiliza para identificar la información del estado (opcional).
- `info`: cadena que se asociará a la notificación de errores para esta transformación (opcional).
- `stageThreshold`: número de errores detectados durante este proceso de transformación que provocarán que el proceso se termine (opcional: cero de forma predeterminada, lo que indica que el proceso no debería terminarse por error).
- `totalThreshold`: número de errores detectados hasta este proceso de transformación incluido que provocarán que el proceso se termine (opcional: cero de forma predeterminada, lo que indica que el proceso no debería terminarse por error).

Por ejemplo, el esquema de una lectura de una exportación con la estructura JSON de DynamoDB puede parecerse al siguiente:

```
root
|-- Item: struct
|    |-- ColA: struct
|    |    |-- S: string
|    |-- ColB: struct
|    |    |-- S: string
|    |-- ColC: struct
|    |    |-- N: string
|    |-- ColD: struct
|    |    |-- L: array
|    |    |    |-- element: null
```

La transformación `unnest_ddb_json()` convertiría esto en:

```
root
|-- ColA: string
|-- ColB: string
|-- ColC: string
|-- ColD: array
|    |-- element: null
```

En el siguiente ejemplo de código, se muestra cómo utilizar el conector de exportación de DynamoDB de AWS Glue, invocar un desanidamiento JSON de DynamoDB e imprimir el número de particiones:

```
import sys
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job
from awsglue.utils import getResolvedOptions

args = getResolvedOptions(sys.argv, ["JOB_NAME"])
glue_context= GlueContext(SparkContext.getOrCreate())
job = Job(glue_context)
job.init(args["JOB_NAME"], args)

dynamicFrame = glue_context.create_dynamic_frame.from_options(
    connection_type="dynamodb",
    connection_options={
```

```
        "dynamodb.export": "ddb",
        "dynamodb.tableArn": "<test_source>",
        "dynamodb.s3.bucket": "<bucket name>",
        "dynamodb.s3.prefix": "<bucket prefix>",
        "dynamodb.s3.bucketOwner": "<account_id>",
    }
)
unnested = dynamicFrame.unnest_ddb_json()
print(unnested.getNumPartitions())

job.commit()
```

write

`write(connection_type, connection_options, format, format_options, accumulator_size)`

Obtiene un DataSink(object) del tipo de conexión especificado del GlueContext clase de este elemento `DynamicFrame` y lo utiliza para escribir y dar formato al contenido de este `DynamicFrame`. Devuelve el nuevo `DynamicFrame` con formato y escrito tal y como se especifica.

- `connection_type`: tipo de conexión que se utilizará. Entre los valores válidos se incluyen: `s3`, `mysql`, `postgresql`, `redshift`, `sqlserver` y `oracle`.
- `connection_options`: opción de conexión que se utilizará (opcional). Para un `connection_type` de `s3`, se define una ruta de Amazon S3.

  ```
  connection_options = {"path": "s3://aws-glue-target/temp"}
  ```

  Para conexiones JDBC, deben definirse varias propiedades. Tenga en cuenta que el nombre de la base de datos debe ser parte de la URL. Opcionalmente, se puede incluir en las opciones de conexión.

  > **Warning**
  >
  > No se recomienda almacenar las contraseñas en el script. Considere utilizar `boto3` para recuperarlas de AWS Secrets Manager o del catálogo de datos de Glue AWS.

```
connection_options = {"url": "jdbc-url/database", "user": "username",
 "password": passwordVariable,"dbtable": "table-name", "redshiftTmpDir": "s3-tempdir-path"}
```

- `format`: una especificación de formato (opcional). Se utiliza para Amazon Simple Storage Service (Amazon S3) o una conexión de AWS Glue que admite diversos formatos. Consulte en Opciones de formato de datos para las entradas y las salidas en AWS Glue para Spark los formatos que se admiten.
- `format_options`: opciones del formato especificado. Consulte en Opciones de formato de datos para las entradas y las salidas en AWS Glue para Spark los formatos que se admiten.
- `accumulator_size`: el tamaño acumulable que se utilizará, en bytes (opcional).

— errores —

- assertErrorThreshold
- errorsAsDynamicFrame
- errorsCount
- stageErrorsCount

assertErrorThreshold

`assertErrorThreshold( )`: confirmación utilizada para los errores de las transformaciones que crearon este elemento `DynamicFrame`. Muestra un valor de `Exception` desde el `DataFrame` subyacente.

errorsAsDynamicFrame

`errorsAsDynamicFrame( )`: devuelve un `DynamicFrame` con registros de error anidados dentro.

Ejemplo: uso de errorsAsDynamicFrame para ver los registros de errores

En el siguiente ejemplo de código se muestra cómo utilizar el método `errorsAsDynamicFrame` para ver un registro de error de un `DynamicFrame`.

Conjunto de datos de ejemplo

El ejemplo utiliza el siguiente conjunto de datos que se puede cargar en Amazon S3 como JSON. Tenga en cuenta que el segundo registro tiene un formato incorrecto. Los datos con formato incorrecto generalmente interrumpen el análisis de archivos cuando se utiliza SparkSQL. Sin embargo, `DynamicFrame` reconoce los problemas de formato incorrecto y convierte las líneas con formato incorrecto en registros de errores que se pueden gestionar de forma individual.

```
{"id": 1, "name": "george", "surname": "washington", "height": 178}
{"id": 2, "name": "benjamin", "surname": "franklin",
{"id": 3, "name": "alexander", "surname": "hamilton", "height": 171}
{"id": 4, "name": "john", "surname": "jay", "height": 190}
```

Código de ejemplo

```
# Example: Use errorsAsDynamicFrame to view error records.
# Replace s3://DOC-EXAMPLE-S3-BUCKET/error_data.json with your location.

from pyspark.context import SparkContext
from awsglue.context import GlueContext

# Create GlueContext
sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

# Create errors DynamicFrame, view schema
errors = glueContext.create_dynamic_frame.from_options(
    "s3", {"paths": ["s3://DOC-EXAMPLE-S3-BUCKET/error_data.json"]}, "json"
)
print("Schema of errors DynamicFrame:")
errors.printSchema()

# Show that errors only contains valid entries from the dataset
print("errors contains only valid records from the input dataset (2 of 4 records)")
errors.toDF().show()

# View errors
print("Errors count:", str(errors.errorsCount()))
print("Errors:")
errors.errorsAsDynamicFrame().toDF().show()

# View error fields and error data
error_record = errors.errorsAsDynamicFrame().toDF().head()
```

```
error_fields = error_record["error"]
print("Error fields: ")
print(error_fields.asDict().keys())

print("\nError record data:")
for key in error_fields.asDict().keys():
    print("\n", key, ": ", str(error_fields[key]))
```

Salida

```
Schema of errors DynamicFrame:
root
|-- id: int
|-- name: string
|-- surname: string
|-- height: int

errors contains only valid records from the input dataset (2 of 4 records)
+---+------+----------+------+
| id|  name|   surname|height|
+---+------+----------+------+
|  1|george|washington|   178|
|  4|  john|       jay|   190|
+---+------+----------+------+

Errors count: 1
Errors:
+--------------------+
|               error|
+--------------------+
|[[  File "/tmp/20...|
+--------------------+

Error fields:
dict_keys(['callsite', 'msg', 'stackTrace', 'input', 'bytesread', 'source',
 'dynamicRecord'])

Error record data:

 callsite :  Row(site='  File "/tmp/2060612586885849088", line 549, in <module>\n
 sys.exit(main())\n  File "/tmp/2060612586885849088", line 523, in main\n    response
 = handler(content)\n  File "/tmp/2060612586885849088", line 197, in execute_request
\n    result = node.execute()\n  File "/tmp/2060612586885849088", line 103, in
```

```
 execute\n    exec(code, global_dict)\n  File "<stdin>", line 10, in <module>\n
 File "/opt/amazon/lib/python3.6/site-packages/awsglue/dynamicframe.py", line 625, in
 from_options\n    format_options, transformation_ctx, push_down_predicate, **kwargs)\n
  File "/opt/amazon/lib/python3.6/site-packages/awsglue/context.py", line 233, in
 create_dynamic_frame_from_options\n    source.setFormat(format, **format_options)\n',
 info='')

 msg :  error in jackson reader

 stackTrace :  com.fasterxml.jackson.core.JsonParseException: Unexpected character
 ('{' (code 123)): was expecting either valid name character (for unquoted name) or
 double-quote (for quoted) to start field name
 at [Source: com.amazonaws.services.glue.readers.BufferedStream@73492578; line: 3,
 column: 2]
 at com.fasterxml.jackson.core.JsonParser._constructError(JsonParser.java:1581)
 at
 com.fasterxml.jackson.core.base.ParserMinimalBase._reportError(ParserMinimalBase.java:533)
 at
 com.fasterxml.jackson.core.base.ParserMinimalBase._reportUnexpectedChar(ParserMinimalBase.java
 at
 com.fasterxml.jackson.core.json.UTF8StreamJsonParser._handleOddName(UTF8StreamJsonParser.java:
 at
 com.fasterxml.jackson.core.json.UTF8StreamJsonParser._parseName(UTF8StreamJsonParser.java:1650
 at
 com.fasterxml.jackson.core.json.UTF8StreamJsonParser.nextToken(UTF8StreamJsonParser.java:740)
 at com.amazonaws.services.glue.readers.JacksonReader$$anonfun$hasNextGoodToken
$1.apply(JacksonReader.scala:57)
 at com.amazonaws.services.glue.readers.JacksonReader$$anonfun$hasNextGoodToken
$1.apply(JacksonReader.scala:57)
 at scala.collection.Iterator$$anon$9.next(Iterator.scala:162)
 at scala.collection.Iterator$$anon$16.hasNext(Iterator.scala:599)
 at scala.collection.Iterator$$anon$16.hasNext(Iterator.scala:598)
 at scala.collection.Iterator$class.foreach(Iterator.scala:891)
 at scala.collection.AbstractIterator.foreach(Iterator.scala:1334)
 at com.amazonaws.services.glue.readers.JacksonReader$$anonfun
$1.apply(JacksonReader.scala:120)
 at com.amazonaws.services.glue.readers.JacksonReader$$anonfun
$1.apply(JacksonReader.scala:116)
 at
 com.amazonaws.services.glue.DynamicRecordBuilder.handleErr(DynamicRecordBuilder.scala:209)
 at
 com.amazonaws.services.glue.DynamicRecordBuilder.handleErrorWithException(DynamicRecordBuilder
 at
 com.amazonaws.services.glue.readers.JacksonReader.nextFailSafe(JacksonReader.scala:116)
```

```
 at com.amazonaws.services.glue.readers.JacksonReader.next(JacksonReader.scala:109)
 at com.amazonaws.services.glue.readers.JSONReader.next(JSONReader.scala:247)
 at
 com.amazonaws.services.glue.hadoop.TapeHadoopRecordReaderSplittable.nextKeyValue(TapeHadoopRec
 at org.apache.spark.rdd.NewHadoopRDD$$anon$1.hasNext(NewHadoopRDD.scala:230)
 at org.apache.spark.InterruptibleIterator.hasNext(InterruptibleIterator.scala:37)
 at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:409)
 at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:409)
 at scala.collection.Iterator$$anon$13.hasNext(Iterator.scala:462)
 at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:409)
 at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:409)
 at scala.collection.Iterator$$anon$13.hasNext(Iterator.scala:462)
 at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:409)
 at scala.collection.Iterator$$anon$11.hasNext(Iterator.scala:409)
 at org.apache.spark.sql.execution.SparkPlan$$anonfun$2.apply(SparkPlan.scala:255)
 at org.apache.spark.sql.execution.SparkPlan$$anonfun$2.apply(SparkPlan.scala:247)
 at org.apache.spark.rdd.RDD$$anonfun$mapPartitionsInternal$1$$anonfun$apply
$24.apply(RDD.scala:836)
 at org.apache.spark.rdd.RDD$$anonfun$mapPartitionsInternal$1$$anonfun$apply
$24.apply(RDD.scala:836)
 at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)
 at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:324)
 at org.apache.spark.rdd.RDD.iterator(RDD.scala:288)
 at org.apache.spark.rdd.MapPartitionsRDD.compute(MapPartitionsRDD.scala:52)
 at org.apache.spark.rdd.RDD.computeOrReadCheckpoint(RDD.scala:324)
 at org.apache.spark.rdd.RDD.iterator(RDD.scala:288)
 at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:90)
 at org.apache.spark.scheduler.Task.run(Task.scala:121)
 at org.apache.spark.executor.Executor$TaskRunner$$anonfun$10.apply(Executor.scala:408)
 at org.apache.spark.util.Utils$.tryWithSafeFinally(Utils.scala:1360)
 at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:414)
 at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
 at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
 at java.lang.Thread.run(Thread.java:750)


 input :

 bytesread :  252

 source :

 dynamicRecord :  Row(id=2, name='benjamin', surname='franklin')
```

errorsCount

`errorsCount( )`: devuelve el número total de errores de un `DynamicFrame`.

stageErrorsCount

`stageErrorsCount`: devuelve el número de errores que se produjo en el proceso de generar este `DynamicFrame`.

Clase DynamicFrameCollection

Un elemento `DynamicFrameCollection` es un diccionario de objetos Clase DynamicFrame, en el que las claves son los nombres de los elementos `DynamicFrames` y los valores son los objetos `DynamicFrame`.

__init__

`__init__(dynamic_frames, glue_ctx)`

- `dynamic_frames`: diccionario de objetos Clase DynamicFrame.
- `glue_ctx`: un objeto GlueContext clase.

Claves

`keys( )`: devuelve una lista de las claves de esta colección, que generalmente se compone de los nombres de los valores de `DynamicFrame` correspondientes.

Valores

`values(key)`: devuelve una lista de los valores de `DynamicFrame` en esta colección.

Seleccionar

`select(key)`

Devuelve el `DynamicFrame` correspondiente a la clave especificada (que, por lo general, es el nombre del elemento `DynamicFrame`).

- `key`: clave del elemento `DynamicFrameCollection` que normalmente representa el nombre de un elemento `DynamicFrame`.

Asignación

`map(callable, transformation_ctx="")`

Utiliza una función que se le ha pasado para crear y devolver un elemento `DynamicFrameCollection` nuevo basado en el elemento `DynamicFrames` de esta colección.

- `callable`: función que toma un elemento `DynamicFrame` y el contexto de transformación especificado como parámetros y devuelve un elemento `DynamicFrame`.
- `transformation_ctx`: contexto de transformación que callable utilizará (opcional).

Flatmap

`flatmap(f, transformation_ctx="")`

Utiliza una función que se le ha pasado para crear y devolver un elemento `DynamicFrameCollection` nuevo basado en el elemento `DynamicFrames` de esta colección.

- `f`: función que toma un elemento `DynamicFrame` como parámetro y devuelve un elemento `DynamicFrame` o `DynamicFrameCollection`.
- `transformation_ctx`: contexto de transformación que la función utilizará (opcional).

Clase DynamicFrameWriter

Métodos

- __init__
- from_options
- from_catalog
- from_jdbc_conf

__init__

`__init__(glue_context)`

- `glue_context`: el GlueContext clase que se va a utilizar.

from_options

`from_options(frame, connection_type, connection_options={}, format=None, format_options={}, transformation_ctx="")`

Escribe un `DynamicFrame` con la conexión y el formato especificados.

- `frame`: el `DynamicFrame` que se va a escribir.
- `connection_type`: el tipo de conexión. Entre los valores válidos se incluyen: `s3`, `mysql`, `postgresql`, `redshift`, `sqlserver` y `oracle`.
- `connection_options`: opciones de conexión, como la tabla de rutas y bases de datos (opcional). Para un `connection_type` de `s3`, se define una ruta de Amazon S3.

```
connection_options = {"path": "s3://aws-glue-target/temp"}
```

Para conexiones JDBC, deben definirse varias propiedades. Tenga en cuenta que el nombre de la base de datos debe ser parte de la URL. Opcionalmente, se puede incluir en las opciones de conexión.

> **Warning**
>
> No se recomienda almacenar las contraseñas en el script. Considere utilizar `boto3` para recuperarlas de AWS Secrets Manager o del catálogo de datos de Glue AWS.

```
connection_options = {"url": "jdbc-url/database", "user": "username",
 "password": passwordVariable,"dbtable": "table-name", "redshiftTmpDir": "s3-tempdir-path"}
```

La propiedad `dbtable` es el nombre de la tabla de JDBC. Para almacenes de datos de JDBC que admiten esquemas dentro de una base de datos, especifique `schema.table-name`. Si no se ha proporcionado un esquema, se usa el esquema "public" predeterminado.

Para obtener más información, consulte Tipos de conexión y opciones para ETL en AWS Glue para Spark.

- `format`: una especificación de formato (opcional). Se utiliza para Amazon Simple Storage Service (Amazon S3) o una conexión de AWS Glue que admite diversos formatos. Consulte en Opciones

de formato de datos para las entradas y las salidas en AWS Glue para Spark los formatos que se admiten.

- `format_options`: opciones del formato especificado. Consulte en Opciones de formato de datos para las entradas y las salidas en AWS Glue para Spark los formatos que se admiten.
- `transformation_ctx`: contexto de transformación que se va a utilizar (opcional).

from_catalog

`from_catalog(frame, name_space, table_name, redshift_tmp_dir="", transformation_ctx="")`

Escribe y devuelve un `DynamicFrame` con la base de datos de catálogos especificada y un nombre de tabla.

- `frame`: el `DynamicFrame` que se va a escribir.
- `name_space`: la base de datos que se va a utilizar.
- `table_name`: el `table_name` que se va a utilizar.
- `redshift_tmp_dir`: directorio provisional de Amazon Redshift que se va a utilizar (opcional).
- `transformation_ctx`: contexto de transformación que se va a utilizar (opcional).
- `additional_options`: opciones adicionales para AWS Glue.

  Para escribir en tablas que se rigen por Lake Formation, puede utilizar estas opciones adicionales:

  - `transactionId` (cadena): ID de transacción en el que se debe escribir en la tabla regida. Esta transacción no se puede confirmar ni anular, de lo contrario, se producirán errores en la escritura.
  - `callDeleteObjectsOnCancel` (booleano, opcional): si se configura en `true` (el valor predeterminado), AWS Glue llama automáticamente a la API `DeleteObjectsOnCancel` después de escribir el objeto en Amazon S3. Para obtener más información, consulte DeleteObjectsOnCancel en la Guía para desarrolladores de AWS Lake Formation.

  Example Ejemplo: Escribir en una tabla regida en Lake Formation

```
txId = glueContext.start_transaction(read_only=False)
glueContext.write_dynamic_frame.from_catalog(
    frame=dyf,
    database = db,
    table_name = tbl,
```

```
    transformation_ctx = "datasource0",
    additional_options={"transactionId":txId})
...
glueContext.commit_transaction(txId)
```

from_jdbc_conf

`from_jdbc_conf(frame, catalog_connection, connection_options={}, redshift_tmp_dir = "", transformation_ctx="")`

Escribe un `DynamicFrame` mediante la información de la conexión de JDBC especificada.

- `frame`: el `DynamicFrame` que se va a escribir.
- `catalog_connection`: conexión al catálogo que se va a utilizar.
- `connection_options`: opciones de conexión, como la tabla de rutas y bases de datos (opcional).
- `redshift_tmp_dir`: directorio provisional de Amazon Redshift que se va a utilizar (opcional).
- `transformation_ctx`: contexto de transformación que se va a utilizar (opcional).

Ejemplo de write_dynamic_frame

En este ejemplo se escribe la salida localmente mediante un `connection_type` de S3 con un argumento de ruta POSIX en `connection_options`, que permite escribir en almacenamiento local.

```
glueContext.write_dynamic_frame.from_options(\
frame = dyf_splitFields,\
connection_options = {'path': '/home/glue/GlueLocalOutput/'},\
connection_type = 's3',\
format = 'json')
```

DynamicFrameReader clase

— métodos —

- __init__
- from_rdd
- from_options

- from_catalog

__init__

`__init__(glue_context)`

- `glue_context`: el GlueContext clase que se va a utilizar.

from_rdd

`from_rdd(data, name, schema=None, sampleRatio=None)`

Lee un `DynamicFrame` en un conjunto de datos distribuido resistente (RDD).

- `data`: el conjunto de datos que se leerá.
- `name`: el nombre donde se efectuará la lectura.
- `schema`: el esquema que se leerá (opcional).
- `sampleRatio`: el ratio de muestra (opcional).

from_options

`from_options(connection_type, connection_options={}, format=None, format_options={}, transformation_ctx="")`

Lee un `DynamicFrame` utilizando la conexión y el formato especificados.

- `connection_type`: el tipo de conexión. Los valores válidos son `s3`, `mysql`, `postgresql`, `redshift`, `sqlserver`, `oracle`, `dynamodb` y `snowflake`.
- `connection_options`: opciones de conexión, como la tabla de rutas y bases de datos (opcional). Para obtener más información, consulte Tipos y opciones de conexión para ETL en AWS Glue for Spark. Para un `connection_type` de `s3`, se definen las rutas de Amazon S3 en una matriz.

  ```
  connection_options = {"paths": [ "s3://mybucket/object_a", "s3://mybucket/object_b"]}
  ```

  Para conexiones JDBC, deben definirse varias propiedades. Tenga en cuenta que el nombre de la base de datos debe ser parte de la URL. Opcionalmente, se puede incluir en las opciones de conexión.

> **Warning**
>
> No se recomienda almacenar las contraseñas en el script. Considere utilizarlos `boto3` para recuperarlos del AWS Secrets Manager catálogo de datos de AWS Glue.

```
connection_options = {"url": "jdbc-url/database", "user": "username",
 "password": passwordVariable,"dbtable": "table-name", "redshiftTmpDir": "s3-tempdir-path"}
```

Para una conexión JDBC que realiza lecturas en paralelo, puede establecer la opción hashfield. Por ejemplo:

```
connection_options = {"url": "jdbc-url/database", "user": "username",
 "password": passwordVariable,"dbtable": "table-name", "redshiftTmpDir": "s3-tempdir-path" , "hashfield": "month"}
```

Para obtener más información, consulte Lectura desde tablas de JDBC en paralelo.

- `format`: una especificación de formato (opcional). Se utiliza para Amazon Simple Storage Service (Amazon S3) o una conexión de AWS Glue que admite diversos formatos. Consulte en Opciones de formato de datos para las entradas y las salidas en AWS Glue para Spark los formatos que se admiten.
- `format_options`: opciones del formato especificado. Consulte en Opciones de formato de datos para las entradas y las salidas en AWS Glue para Spark los formatos que se admiten.
- `transformation_ctx`: contexto de transformación que se va a utilizar (opcional).
- `push_down_predicate`: filtra particiones sin tener que enumerar y leer todos los archivos del conjunto de datos. Para obtener más información, consulte Filtrado previo con predicados de inserción.

from_catalog

`from_catalog(database, table_name, redshift_tmp_dir="", transformation_ctx="", push_down_predicate="", additional_options={})`

Lee un `DynamicFrame` utilizando el espacio de nombres del catálogo y un nombre de tabla.

- `database`: la base de datos de lectura.
- `table_name`: nombre de la tabla de lectura.
- `redshift_tmp_dir`: un directorio de Amazon Redshift provisorio que se usará (opcional si no se leen datos de Redshift).
- `transformation_ctx`: contexto de transformación que se va a utilizar (opcional).
- `push_down_predicate`: filtra particiones sin tener que enumerar y leer todos los archivos del conjunto de datos. Para obtener más información, consulte Filtrado previo con predicados de inserción.
- `additional_options`: opciones adicionales para AWS Glue.
  - Para usar una conexión JDBC que realiza lecturas en paralelo, puede establecer las opciones `hashfield`, `hashexpression` o `hashpartitions`. Por ejemplo:

    ```
    additional_options = {"hashfield": "month"}
    ```

    Para obtener más información, consulte Lectura desde tablas de JDBC en paralelo.
  - Para transferir una expresión de catálogo para filtrar en función de las columnas de índice, puede ver la opción `catalogPartitionPredicate`.

    `catalogPartitionPredicate`: puede transferir una expresión de catálogo para filtrar en función de las columnas de índice. Esto inserta el filtrado hacia el lado del servidor. Para obtener más información, consulte Índices de partición de AWS Glue. Tenga en cuenta que `push_down_predicate` y `catalogPartitionPredicate` utilizan sintaxis diferentes. El primero utiliza la sintaxis estándar de Spark SQL y el segundo utiliza el analizador JSQL.

    Para obtener más información, consulte Administración de particiones para la salida de ETL en AWS Glue.

GlueContext clase

Envuelve el SparkContextobjeto Apache Spark y, por lo tanto, proporciona mecanismos para interactuar con la plataforma Apache Spark.

__init__

`__init__(sparkContext)`

- `sparkContext`: el contexto de Apache Spark que se va a utilizar.

Creación

- __init__
- getSource
- create_dynamic_frame_from_rdd
- create_dynamic_frame_from_catalog
- create_dynamic_frame_from_options
- create_sample_dynamic_frame_from_catalog
- create_sample_dynamic_frame_from_options
- add_ingestion_time_columns
- create_data_frame_from_catalog
- create_data_frame_from_options
- forEachBatch

getSource

`getSource(connection_type, transformation_ctx = "", **options)`

Crea un objeto `DataSource` que se puede utilizar para leer `DynamicFrames` desde fuentes externas.

- `connection_type`: el tipo de conexión que se va a utilizar, como Amazon Simple Storage Service (Amazon S3), Amazon Redshift y JDBC. Los valores válidos son `s3`, `mysql`, `postgresql`, `redshift`, `sqlserver`, `oracle` y `dynamodb`.
- `transformation_ctx`: contexto de transformación que se va a utilizar (opcional).
- `options`: conjunto de pares nombre-valor opcionales. Para obtener más información, consulte Tipos de conexión y opciones para ETL en AWS Glue para Spark.

A continuación, se muestra un ejemplo de cómo se utiliza `getSource`.

```
>>> data_source = context.getSource("file", paths=["/in/path"])
>>> data_source.setFormat("json")
>>> myFrame = data_source.getFrame()
```

create_dynamic_frame_from_rdd

`create_dynamic_frame_from_rdd(data, name, schema=None, sample_ratio=None, transformation_ctx="")`

Muestra un `DynamicFrame` que se crea a partir de un conjunto de datos distribuido resistente (RDD) de Apache Spark.

- `data`: el origen de datos que se va a utilizar.
- `name`: el nombre de los datos que se van a utilizar.
- `schema`: el esquema que se va a utilizar (opcional).
- `sample_ratio`: el ratio de muestra que se va a utilizar (opcional).
- `transformation_ctx`: contexto de transformación que se va a utilizar (opcional).

create_dynamic_frame_from_catalog

`create_dynamic_frame_from_catalog(database, table_name, redshift_tmp_dir, transformation_ctx = "", push_down_predicate= "", additional_options = {}, catalog_id = None)`

Muestra un `DynamicFrame` que se crea mediante una base de datos del Catálogo de datos y un nombre de tabla. Al usar este método, se proporcionan `format_options` las propiedades de la tabla AWS Glue Data Catalog especificada y otras opciones a través del `additional_options` argumento.

- `Database`: la base de datos de lectura.
- `table_name`: nombre de la tabla de lectura.
- `redshift_tmp_dir`: directorio provisional de Amazon Redshift que se va a utilizar (opcional).
- `transformation_ctx`: contexto de transformación que se va a utilizar (opcional).
- `push_down_predicate`: filtra particiones sin tener que enumerar y leer todos los archivos del conjunto de datos. Para conocer las fuentes compatibles y las limitaciones, consulta Optimizar las lecturas con pulsaciones en AWS Glue ETL. Para obtener más información, consulte Filtrado previo con predicados de inserción.
- `additional_options`: conjunto de pares nombre-valor opcionales. Las opciones posibles incluyen las enumeradas en Tipos de conexión y opciones para ETL en AWS Glue para Spark, excepto por `endpointUrl`, `streamName`, `bootstrap.servers`,

`security.protocol`, `topicName`, `classification` y `delimiter`. Otra opción soportada es `catalogPartitionPredicate`:

`catalogPartitionPredicate`: puede transferir una expresión de catálogo para filtrar en función de las columnas de índice. Esto inserta el filtrado hacia el lado del servidor. Para obtener más información, consulte Índices de partición de AWS Glue. Tenga en cuenta que `push_down_predicate` y `catalogPartitionPredicate` utilizan sintaxis diferentes. El primero utiliza la sintaxis estándar de Spark SQL y el segundo utiliza el analizador JSQL.

- `catalog_id`: ID de catálogo (ID de cuenta) del Catálogo de datos al que se accede. Cuando el valor es Ninguno, se utiliza el ID de cuenta predeterminado del intermediario.

create_dynamic_frame_from_options

`create_dynamic_frame_from_options(connection_type, connection_options={}, format=None, format_options={}, transformation_ctx = "")`

Muestra un `DynamicFrame` que se crea con la conexión y el formato especificados.

- `connection_type`: el tipo de conexión, como Amazon S3, Amazon Redshift y JDBC. Los valores válidos son `s3`, `mysql`, `postgresql`, `redshift`, `sqlserver`, `oracle` y `dynamodb`.
- `connection_options`: las opciones de conexión, como las rutas y la tabla de bases de datos (opcional). Para un `connection_type` de `s3`, se define una lista de rutas de Amazon S3.

```
connection_options = {"paths": ["s3://aws-glue-target/temp"]}
```

Para conexiones JDBC, deben definirse varias propiedades. Tenga en cuenta que el nombre de la base de datos debe ser parte de la URL. Opcionalmente, se puede incluir en las opciones de conexión.

> **Warning**
>
> No se recomienda almacenar las contraseñas en el script. Considere utilizarlos `boto3` para recuperarlos del AWS Secrets Manager catálogo de datos de AWS Glue.

```
connection_options = {"url": "jdbc-url/database", "user": "username",
 "password": passwordVariable,"dbtable": "table-name", "redshiftTmpDir": "s3-tempdir-
path"}
```

La propiedad `dbtable` es el nombre de la tabla de JDBC. Para almacenes de datos de JDBC que admiten esquemas dentro de una base de datos, especifique `schema.table-name`. Si no se ha proporcionado un esquema, se usa el esquema "public" predeterminado.

Para obtener más información, consulte Tipos de conexión y opciones para ETL en AWS Glue para Spark.

- `format`— Una especificación de formato. Se utiliza con una conexión de Amazon S3 o AWS Glue que admite diversos formatos. Consulte en Opciones de formato de datos para las entradas y las salidas en AWS Glue para Spark los formatos que se admiten.
- `format_options`: opciones del formato especificado. Consulte en Opciones de formato de datos para las entradas y las salidas en AWS Glue para Spark los formatos que se admiten.
- `transformation_ctx`: contexto de transformación que se va a utilizar (opcional).
- `push_down_predicate`: filtra particiones sin tener que enumerar y leer todos los archivos del conjunto de datos. Para conocer las fuentes compatibles y las limitaciones, consulta Optimizar las lecturas con pulsaciones en AWS Glue ETL. Para obtener más información, consulte Filtrado previo con predicados de inserción.

create_sample_dynamic_frame_from_catalog

`create_sample_dynamic_frame_from_catalog(database, table_name, num, redshift_tmp_dir, transformation_ctx = "", push_down_predicate= "", additional_options = {}, sample_options = {}, catalog_id = None)`

Muestra un `DynamicFrame` de ejemplo que se crea mediante una base de datos del Catálogo de datos y un nombre de tabla. La `DynamicFrame` solo contiene los primeros registros de `num` de un origen de datos.

- `database`: la base de datos de lectura.
- `table_name`: nombre de la tabla de lectura.
- `num`: número máximo de registros del marco dinámico de muestra arrojado.
- `redshift_tmp_dir`: directorio provisional de Amazon Redshift que se va a utilizar (opcional).

- `transformation_ctx`: contexto de transformación que se va a utilizar (opcional).
- `push_down_predicate`: filtra particiones sin tener que enumerar y leer todos los archivos del conjunto de datos. Para obtener más información, consulte Filtrado previo con predicados de inserción.
- `additional_options`: conjunto de pares nombre-valor opcionales. Las opciones posibles incluyen las enumeradas en Tipos de conexión y opciones para ETL en AWS Glue para Spark, excepto por `endpointUrl`, `streamName`, `bootstrap.servers`, `security.protocol`, `topicName`, `classification` y `delimiter`.
- `sample_options`: parámetros para controlar el comportamiento del muestreo (opcional). Parámetros disponibles actuales para los orígenes de Simple Storage Service (Amazon S3):
  - `maxSamplePartitions`: número máximo de particiones que leerá el muestreo. El valor predeterminado es 10
  - `maxSampleFilesPerPartition`: número máximo de archivos que leerá el muestreo en una partición. El valor predeterminado es 10.

    Estos parámetros ayudan a reducir el tiempo que consume el listado de archivos. Por ejemplo, supongamos que el conjunto de datos tiene 1000 particiones y cada partición tiene 10 archivos. Si se configura `maxSamplePartitions` = 10 y `maxSampleFilesPerPartition` = 10, en lugar de enumerar los 10 000 archivos, el muestreo solo mostrará y leerá las 10 primeras particiones con los 10 primeros archivos de cada uno: 10 * 10 = 100 archivos en total.
- `catalog_id`: el ID de catálogo del Catálogo de datos al que se accede (el ID de cuenta del Catálogo de datos). De forma predeterminada, se establece en `None`. `None` es el valor predeterminado para el ID de catálogo de la cuenta del servicio que hace la llamada.

create_sample_dynamic_frame_from_options

`create_sample_dynamic_frame_from_options(connection_type, connection_options={}, num, sample_options={}, format=None, format_options={}, transformation_ctx = "")`

Arroja un `DynamicFrame` de ejemplo que se crea con la conexión y el formato especificados. La `DynamicFrame` solo contiene los primeros registros de `num` de un origen de datos.

- `connection_type`: el tipo de conexión, como Amazon S3, Amazon Redshift y JDBC. Los valores válidos son `s3`, `mysql`, `postgresql`, `redshift`, `sqlserver`, `oracle` y `dynamodb`.

- `connection_options`: las opciones de conexión, como las rutas y la tabla de bases de datos (opcional). Para obtener más información, consulte Tipos de conexión y opciones para ETL en AWS Glue para Spark.
- `num`: número máximo de registros del marco dinámico de muestra arrojado.
- `sample_options`: parámetros para controlar el comportamiento del muestreo (opcional). Parámetros disponibles actuales para los orígenes de Simple Storage Service (Amazon S3):
  - `maxSamplePartitions`: número máximo de particiones que leerá el muestreo. El valor predeterminado es 10
  - `maxSampleFilesPerPartition`: número máximo de archivos que leerá el muestreo en una partición. El valor predeterminado es 10.

    Estos parámetros ayudan a reducir el tiempo que consume el listado de archivos. Por ejemplo, supongamos que el conjunto de datos tiene 1000 particiones y cada partición tiene 10 archivos. Si se configura `maxSamplePartitions` = 10 y `maxSampleFilesPerPartition` = 10, en lugar de enumerar los 10 000 archivos, el muestreo solo mostrará y leerá las 10 primeras particiones con los 10 primeros archivos de cada uno: 10 * 10 = 100 archivos en total.
- `format`— Una especificación de formato. Se utiliza con una conexión de Amazon S3 o AWS Glue que admite diversos formatos. Consulte en Opciones de formato de datos para las entradas y las salidas en AWS Glue para Spark los formatos que se admiten.
- `format_options`: opciones del formato especificado. Consulte en Opciones de formato de datos para las entradas y las salidas en AWS Glue para Spark los formatos que se admiten.
- `transformation_ctx`: contexto de transformación que se va a utilizar (opcional).
- `push_down_predicate`: filtra particiones sin tener que enumerar y leer todos los archivos del conjunto de datos. Para obtener más información, consulte Filtrado previo con predicados de inserción.

add_ingestion_time_columns

`add_ingestion_time_columns(dataFrame, timeGranularity = "")`

Agrega columnas de tiempo de ingesta como `ingest_year`, `ingest_month`, `ingest_day`, `ingest_hour`, `ingest_minute` al `DataFrame` de entrada. Esta función se genera en forma automática en el script generado por AWS Glue cuando especifique una tabla del Catálogo de datos con Amazon S3 como destino. Esta función actualiza en forma automática la partición con columnas de tiempo de ingesta en la tabla de salida. Esto permite que los datos de salida se dividan

automáticamente en el tiempo de ingesta sin requerir columnas de tiempo de ingesta explícitas en los datos de entrada.

- `dataFrame`: el `dataFrame` al que anexar las columnas de tiempo de ingesta.
- `timeGranularity`: la granularidad de las columnas de tiempo. Los valores válidos son "`day`", "`hour`" y "`minute`". Por ejemplo, si "`hour`" se transfiere a la función, el `dataFrame` original tendrá las columnas de tiempo "`ingest_year`," "`ingest_month`," "`ingest_day`" y "`ingest_hour`" anexadas.

Devuelve el marco de datos después de anexar las columnas de granularidad de tiempo.

Ejemplo:

```
dynamic_frame = DynamicFrame.fromDF(glueContext.add_ingestion_time_columns(dataFrame,
 "hour"))
```

create_data_frame_from_catalog

`create_data_frame_from_catalog(database, table_name, transformation_ctx = "", additional_options = {})`

Devuelve un `DataFrame` que se crea con información de una tabla del Catálogo de datos.

- `database`: la base de datos del Catálogo de datos de la que se va a leer.
- `table_name`: el nombre de la tabla del Catálogo de datos de la que se va a leer.
- `transformation_ctx`: contexto de transformación que se va a utilizar (opcional).
- `additional_options`: conjunto de pares nombre-valor opcionales. Las opciones posibles incluyen las enumeradas en Tipos de conexión y opciones para ETL en AWS Glue para Spark para orígenes de streaming, como `startingPosition`, `maxFetchTimeInMs` y `startingOffsets`.
  - `useSparkDataSource`— Si se establece en true, AWS Glue debe usar la API nativa de Spark Data Source para leer la tabla. La API de orígenes de datos de Spark admite los siguientes formatos: AVRO, binario, CSV, JSON, ORC, Parquet y texto. En una tabla del Catálogo de datos, especifique el formato mediante la propiedad `classification`. Para obtener más información sobre la API de orígenes de datos de Spark, consulte la documentación oficial de Apache Spark.

El uso de `create_data_frame_from_catalog` con `useSparkDataSource` presenta las siguientes ventajas:

- Devuelve directamente un `DataFrame` y ofrece una alternativa a `create_dynamic_frame.from_catalog().toDF()`.
- Admite el control AWS Lake Formation de permisos a nivel de tabla para los formatos nativos.
- Admite la lectura de formatos de lagos de datos sin control de permisos a AWS Lake Formation nivel de tabla. Para obtener más información, consulte Uso de marcos de lagos de datos con trabajos de ETL de AWS Glue.

Cuando lo habilitas`useSparkDataSource`, también puedes añadir cualquiera de las opciones de fuente de datos de Spark `additional_options` según sea necesario. AWS Glue pasa estas opciones directamente al lector Spark.

- `useCatalogSchema`— Cuando se establece en true, AWS Glue aplica el esquema del catálogo de datos al resultado`DataFrame`. De lo contrario, el lector deduce el esquema a partir de los datos. Cuando se habilita `useCatalogSchema`, también se debe establecer `useSparkDataSource` en true (verdadero).

Limitaciones

Tenga en cuenta las siguientes limitaciones cuando utilice la opción `useSparkDataSource`:

- Cuando lo usas`useSparkDataSource`, AWS Glue crea una nueva `DataFrame` sesión de Spark distinta de la sesión original de Spark.
- El filtrado de DataFrame particiones de Spark no funciona con las siguientes funciones de AWS Glue.
  - Marcadores de trabajo
  - Exclusión de clases de almacenamiento de Amazon S3
  - Predicados de particiones de catálogo

  Para utilizar el filtrado de particiones con estas funciones, puedes usar el predicado desplegable AWS Glue. Para obtener más información, consulte Filtrado previo con predicados de inserción. El filtrado de columnas no particionadas no se ve afectado.

  En el siguiente script de ejemplo, se muestra la forma incorrecta de filtrar particiones con la opción `excludeStorageClasses`.

```
// Incorrect partition filtering using Spark filter with excludeStorageClasses
read_df = glueContext.create_data_frame.from_catalog(
    database=database_name,
    table_name=table_name,
    additional_options = {
      "useSparkDataSource": True,
      "excludeStorageClasses" : ["GLACIER", "DEEP_ARCHIVE"]
    }
)

//  Suppose year and month are partition keys.
//  Filtering on year and month won't work, the filtered_df will still
//  contain data with other year/month values.
filtered_df = read_df.filter("year == '2017 and month == '04' and 'state == 'CA'")
```

En el siguiente script de ejemplo, se muestra la forma correcta de utilizar un predicado de inserción para filtrar particiones con la opción `excludeStorageClasses`.

```
// Correct partition filtering using the AWS Glue pushdown predicate
// with excludeStorageClasses
read_df = glueContext.create_data_frame.from_catalog(
    database=database_name,
    table_name=table_name,
    //  Use AWS Glue pushdown predicate to perform partition filtering
    push_down_predicate = "(year=='2017' and month=='04')"
    additional_options = {
      "useSparkDataSource": True,
      "excludeStorageClasses" : ["GLACIER", "DEEP_ARCHIVE"]
    }
)

//  Use Spark filter only on non-partitioned columns
filtered_df = read_df.filter("state == 'CA'")
```

Ejemplo: crear una tabla CSV con el lector de orígenes de datos de Spark

```
//  Read a CSV table with '\t' as separator
read_df = glueContext.create_data_frame.from_catalog(
    database=<database_name>,
    table_name=<table_name>,
```

```
    additional_options = {"useSparkDataSource": True,  "sep": '\t'}
)
```

create_data_frame_from_options

`create_data_frame_from_options(connection_type, connection_options={}, format=None, format_options={}, transformation_ctx = "")`

Esta API ha quedado obsoleta. En su lugar, use la API `getSource()`. Muestra un `DataFrame` que se crea con la conexión y el formato especificados. Utilice esta función solo con orígenes de streaming de AWS Glue.

- `connection_type`: el tipo de conexión de streaming. Los valores válidos son `kinesis` y `kafka`.
- `connection_options`: opciones de conexión, que son diferentes para Kinesis y Kafka. Puede encontrar la lista de todas las opciones de conexión para cada origen de datos de streaming en Tipos de conexión y opciones para ETL en AWS Glue para Spark. Tenga en cuenta las siguientes diferencias en las opciones de conexión de streaming:
  - Los orígenes de streaming de Kinesis requieren `streamARN`, `startingPosition`, `inferSchema` y `classification`.
  - Los orígenes de streaming de Kafka requieren `connectionName`, `topicName`, `startingOffsets`, `inferSchema` y `classification`.
- `format`— Una especificación de formato. Se utiliza con una conexión de Amazon S3 o AWS Glue que admite diversos formatos. Para obtener información acerca de los formatos soportados, consulte Opciones de formato de datos para las entradas y las salidas en AWS Glue para Spark.
- `format_options`: opciones del formato especificado. Para obtener información acerca de las opciones de formatos soportados, consulte Opciones de formato de datos para las entradas y las salidas en AWS Glue para Spark.
- `transformation_ctx`: contexto de transformación que se va a utilizar (opcional).

Ejemplo de origen de streaming de Amazon Kinesis:

```
kinesis_options =
   { "streamARN": "arn:aws:kinesis:us-east-2:777788889999:stream/fromOptionsStream",
     "startingPosition": "TRIM_HORIZON",
     "inferSchema": "true",
     "classification": "json"
   }
```

```
data_frame_datasource0 =
 glueContext.create_data_frame.from_options(connection_type="kinesis",
 connection_options=kinesis_options)
```

Ejemplo de origen de streaming de Kafka:

```
kafka_options =
    { "connectionName": "ConfluentKafka",
      "topicName": "kafka-auth-topic",
      "startingOffsets": "earliest",
      "inferSchema": "true",
      "classification": "json"
    }
data_frame_datasource0 =
 glueContext.create_data_frame.from_options(connection_type="kafka",
 connection_options=kafka_options)
```

forEachBatch

`forEachBatch(frame, batch_function, options)`

Se aplica la `batch_function` transferida a cada microlote que se lee desde el origen de streaming.

- `frame`— El DataFrame que contiene el microlote actual.
- `batch_function`: una función que se aplicará para cada microlote.
- `options`: una recopilación de pares clave-valor que contiene información sobre cómo procesar microlotes. Se requieren las siguientes opciones:
  - `windowSize`: cantidad de tiempo que se debe dedicar al procesamiento de cada lote.
  - `checkpointLocation`: la ubicación donde se almacenan los puntos de verificación para el trabajo de ETL de streaming.
  - `batchMaxRetries`: número máximo de reintentos permitidos para este lote si se genera un error. El valor predeterminado es 3. Esta opción sólo se puede configurar para Glue versión 2.0 y superior.

Ejemplo:

```
glueContext.forEachBatch(
    frame = data_frame_datasource0,
```

```
    batch_function = processBatch,
    options = {
        "windowSize": "100 seconds",
        "checkpointLocation": "s3://kafka-auth-dataplane/confluent-test/output/
checkpoint/"
    }
)

def processBatch(data_frame, batchId):
    if (data_frame.count() > 0):
        datasource0 = DynamicFrame.fromDF(
          glueContext.add_ingestion_time_columns(data_frame, "hour"),
          glueContext, "from_data_frame"
        )
        additionalOptions_datasink1 = {"enableUpdateCatalog": True}
        additionalOptions_datasink1["partitionKeys"] = ["ingest_yr", "ingest_mo",
 "ingest_day"]
        datasink1 = glueContext.write_dynamic_frame.from_catalog(
          frame = datasource0,
          database = "tempdb",
          table_name = "kafka-auth-table-output",
          transformation_ctx = "datasink1",
          additional_options = additionalOptions_datasink1
        )
```

Trabajar con conjuntos de datos en Amazon S3

- purge_table
- purge_s3_path
- transition_table
- transition_s3_path

purge_table

purge_table(catalog_id=None, database="", table_name="", options={}, transformation_ctx="")

Elimina archivos de Amazon S3 correspondientes a la base de datos y la tabla del catálogo especificado. Si se eliminan todos los archivos de una partición, esa partición también se eliminará del catálogo.

Si desea poder recuperar los objetos eliminados, puede habilitar control de versiones de objetos en el bucket de Amazon S3. Cuando se elimina un objeto de un bucket que no tiene habilitado el control de versiones de objetos, el objeto no se puede recuperar. Para obtener más información acerca de cómo recuperar objetos eliminados en un bucket habilitado para versiones, consulte ¿Cómo puedo recuperar un objeto de Amazon S3 eliminado? en el Centro de conocimientos de AWS Support .

- `catalog_id`: el ID de catálogo del Catálogo de datos al que se accede (el ID de cuenta del Catálogo de datos). De forma predeterminada, se establece en `None`. `None` es el valor predeterminado para el ID de catálogo de la cuenta del servicio que hace la llamada.
- `database`: la base de datos que se va a utilizar.
- `table_name`: nombre de la tabla que se utilizará.
- `options`: opciones para filtrar los archivos que se van a eliminar y para la generación de archivos de manifiesto.
  - `retentionPeriod`: especifica un período para retener los archivos en cantidad de horas. Se mantienen los archivos que son posteriores al período de retención. De forma predeterminada, se establece en 168 horas (7 días).
  - `partitionPredicate`: se eliminan las particiones que cumplen con este predicado. Los archivos comprendidos en el período de retención de estas particiones no se eliminan. Configurar en `""`, valor vacío de forma predeterminada.
  - `excludeStorageClasses`: no se eliminan los archivos con clase de almacenamiento configurada en `excludeStorageClasses`. El valor predeterminado es `Set()`, un conjunto vacío.
  - `manifestFilePath`: una ruta opcional para la generación de archivos de manifiesto. Todos los archivos que se depuraron correctamente se registran en `Success.csv`, mientras que los que no lo hicieron se registran en `Failed.csv`
- `transformation_ctx`: contexto de transformación que se va a utilizar (opcional). Se utiliza en la ruta del archivo de manifiesto.

Example

```
glueContext.purge_table("database", "table", {"partitionPredicate": "(month=='march')",
 "retentionPeriod": 1, "excludeStorageClasses": ["STANDARD_IA"], "manifestFilePath":
 "s3://bucketmanifest/"})
```

purge_s3_path

`purge_s3_path(s3_path, options={}, transformation_ctx="")`

Elimina archivos de la ruta de Amazon S3 especificada recursivamente.

Si desea poder recuperar los objetos eliminados, puede habilitar control de versiones de objetos en el bucket de Amazon S3. Cuando se elimina un objeto de un bucket que no tiene habilitado el control de versiones de objetos, el objeto no se puede recuperar. Para obtener más información sobre cómo recuperar objetos eliminados en un bucket con control de versiones, consulte ¿Cómo puedo recuperar un objeto de Amazon S3 que se eliminó? en el Centro de AWS Support conocimiento.

- `s3_path`: la ruta de acceso en Amazon S3 de los archivos que se van a eliminar en el formato `s3://<bucket>/<prefix>/`
- `options`: opciones para filtrar los archivos que se van a eliminar y para la generación de archivos de manifiesto.
  - `retentionPeriod`: especifica un período para retener los archivos en cantidad de horas. Se mantienen los archivos que son posteriores al período de retención. De forma predeterminada, se establece en 168 horas (7 días).
  - `excludeStorageClasses`: no se eliminan los archivos con clase de almacenamiento configurada en `excludeStorageClasses`. El valor predeterminado es `Set()`, un conjunto vacío.
  - `manifestFilePath`: una ruta opcional para la generación de archivos de manifiesto. Todos los archivos que se depuraron correctamente se registran en `Success.csv`, mientras que los que no lo hicieron se registran en `Failed.csv`
- `transformation_ctx`: contexto de transformación que se va a utilizar (opcional). Se utiliza en la ruta del archivo de manifiesto.

Example

```
glueContext.purge_s3_path("s3://bucket/path/", {"retentionPeriod": 1,
 "excludeStorageClasses": ["STANDARD_IA"], "manifestFilePath": "s3://bucketmanifest/"})
```

transition_table

`transition_table(database, table_name, transition_to, options={}, transformation_ctx="", catalog_id=None)`

Inicia la transición de la clase de almacenamiento de los archivos almacenados en Amazon S3 correspondientes a la base de datos y la tabla del catálogo especificado.

Puede realizar la transición entre dos clases de almacenamiento cualquiera. En el caso de las clases de almacenamiento `GLACIER` y `DEEP_ARCHIVE`, la transición puede hacerse a estas clases. Sin embargo, debería utilizar `S3 RESTORE` para realizar la transición de `GLACIER` y las clases de almacenamiento `DEEP_ARCHIVE`.

Si ejecuta trabajos de ETL de AWS Glue que leen archivos o particiones de Amazon S3, puede excluir algunos tipos de clases de almacenamiento de Amazon S3. Para obtener más información, consulte Exclusión de clases de almacenamiento de Amazon S3.

- `database`: la base de datos que se va a utilizar.
- `table_name`: nombre de la tabla que se utilizará.
- `transition_to`: la clase de almacenamiento de Amazon S3 hacia la que se hará la transición.
- `options`: opciones para filtrar los archivos que se van a eliminar y para la generación de archivos de manifiesto.
  - `retentionPeriod`: especifica un período para retener los archivos en cantidad de horas. Se mantienen los archivos que son posteriores al período de retención. De forma predeterminada, se establece en 168 horas (7 días).
  - `partitionPredicate`: se realiza la transición de las particiones que cumplen con este predicado. Los archivos comprendidos en el período de retención de estas particiones no realizan la transición. Configurar en `""`, valor vacío de forma predeterminada.
  - `excludeStorageClasses`: no se realiza la transición de los archivos con clase de almacenamiento en `excludeStorageClasses`. El valor predeterminado es `Set()`, un conjunto vacío.
  - `manifestFilePath`: una ruta opcional para la generación de archivos de manifiesto. Todos los archivos que realizaron la transición correctamente se registran en `Success.csv`, mientras que los que no lo hicieron se registran en `Failed.csv`
  - `accountId`: el ID de cuenta de Amazon Web Services para ejecutar la transformación de transición. Es obligatorio para esta transformación.

  - `roleArn`— La AWS función de ejecutar la transformación de la transición. Es obligatorio para esta transformación.
- `transformation_ctx`: contexto de transformación que se va a utilizar (opcional). Se utiliza en la ruta del archivo de manifiesto.
- `catalog_id`: el ID de catálogo del Catálogo de datos al que se accede (el ID de cuenta del Catálogo de datos). De forma predeterminada, se establece en `None`. `None` es el valor predeterminado para el ID de catálogo de la cuenta del servicio que hace la llamada.

Example

```
glueContext.transition_table("database", "table", "STANDARD_IA", {"retentionPeriod": 1,
 "excludeStorageClasses": ["STANDARD_IA"], "manifestFilePath": "s3://bucketmanifest/",
 "accountId": "12345678901", "roleArn": "arn:aws:iam::123456789012:user/example-
username"})
```

transition_s3_path

`transition_s3_path(s3_path, transition_to, options={}, transformation_ctx="")`

Realiza transiciones recursivas en la clase de almacenamiento de los archivos de la ruta de Amazon S3 especificada.

Puede realizar la transición entre dos clases de almacenamiento cualquiera. En el caso de las clases de almacenamiento `GLACIER` y `DEEP_ARCHIVE`, la transición puede hacerse a estas clases. Sin embargo, debería utilizar `S3 RESTORE` para realizar la transición de `GLACIER` y las clases de almacenamiento `DEEP_ARCHIVE`.

Si ejecuta trabajos de ETL de AWS Glue que leen archivos o particiones de Amazon S3, puede excluir algunos tipos de clases de almacenamiento de Amazon S3. Para obtener más información, consulte Exclusión de clases de almacenamiento de Amazon S3.

- `s3_path`: la ruta en Amazon S3 de los archivos sobre los que se realizará la transición en el formato `s3://<bucket>/<prefix>/`
- `transition_to`: la clase de almacenamiento de Amazon S3 hacia la que se hará la transición.
- `options`: opciones para filtrar los archivos que se van a eliminar y para la generación de archivos de manifiesto.

- `retentionPeriod`: especifica un período para retener los archivos en cantidad de horas. Se mantienen los archivos que son posteriores al período de retención. De forma predeterminada, se establece en 168 horas (7 días).
- `partitionPredicate`: se realiza la transición de las particiones que cumplen con este predicado. Los archivos comprendidos en el período de retención de estas particiones no realizan la transición. Configurar en `""`, valor vacío de forma predeterminada.
- `excludeStorageClasses`: no se realiza la transición de los archivos con clase de almacenamiento en `excludeStorageClasses`. El valor predeterminado es `Set()`, un conjunto vacío.
- `manifestFilePath`: una ruta opcional para la generación de archivos de manifiesto. Todos los archivos que realizaron la transición correctamente se registran en `Success.csv`, mientras que los que no lo hicieron se registran en `Failed.csv`
- `accountId`: el ID de cuenta de Amazon Web Services para ejecutar la transformación de transición. Es obligatorio para esta transformación.
- `roleArn`— La AWS función de ejecutar la transformación de transición. Es obligatorio para esta transformación.

- `transformation_ctx`: contexto de transformación que se va a utilizar (opcional). Se utiliza en la ruta del archivo de manifiesto.

Example

```
glueContext.transition_s3_path("s3://bucket/prefix/", "STANDARD_IA",
 {"retentionPeriod": 1, "excludeStorageClasses": ["STANDARD_IA"],
 "manifestFilePath": "s3://bucketmanifest/", "accountId": "12345678901", "roleArn":
 "arn:aws:iam::123456789012:user/example-username"})
```

Extracción

- extract_jdbc_conf

extract_jdbc_conf

`extract_jdbc_conf(connection_name, catalog_id = None)`

Devuelve `dict` con claves con las propiedades de configuración del objeto de conexión de AWS Glue en el Catálogo de datos.

- `user`: nombre de usuario de la base de datos.
- `password`: contraseña de la base de datos.
- `vendor`: especifica un proveedor (`mysql`, `postgresql`, `oracle`, `sqlserver`, etc.).
- `enforceSSL`: una cadena booleana que indica si se requiere una conexión segura.
- `customJDBCCert`: uso de un certificado de cliente específico de la ruta de Amazon S3 indicada.
- `skipCustomJDBCCertValidation`: una cadena booleana que indica si `customJDBCCert` debe ser validado por una CA.
- `customJDBCCertString`: información adicional sobre el certificado personalizado, específico para el tipo de controlador.
- `url`: URL de JDBC (obsoleta) con solo protocolo, servidor y puerto.
- `fullUrl`: URL de JDBC introducida cuando se creó la conexión (Disponible en AWS Glue versión 3.0 o posterior).

Ejemplo de recuperación de configuraciones de JDBC:

```
jdbc_conf = glueContext.extract_jdbc_conf(connection_name="your_glue_connection_name")
print(jdbc_conf)
>>> {'enforceSSL': 'false', 'skipCustomJDBCCertValidation': 'false', 'url':
 'jdbc:mysql://myserver:3306', 'fullUrl': 'jdbc:mysql://myserver:3306/mydb',
 'customJDBCCertString': '', 'user': 'admin', 'customJDBCCert': '', 'password': '1234',
 'vendor': 'mysql'}
```

Transacciones

- start_transaction
- commit_transaction
- cancel_transaction

start_transaction

`start_transaction(read_only)`

Inicie una nueva transacción. Llama de forma interna a la API startTransaction de Lake Formation.

- `read_only`: (booleano) indica si esta transacción debe ser de solo lectura o de lectura y escritura. Se rechazarán las escrituras realizadas con un ID de transacción de solo lectura. No es necesario confirmar las transacciones de solo lectura.

Devuelve el ID de la transacción.

commit_transaction

`commit_transaction(transaction_id, wait_for_commit = True)`

Intenta confirmar la transacción especificada. Es posible que se devuelva `commit_transaction` antes de que la transacción haya terminado de confirmarse. Llama de forma interna a la API commitTransaction de Lake Formation.

- `transaction_id` : (cadena) la transacción que se confirmará.
- `wait_for_commit`: (booleano) determina si se devuelve `commit_transaction` de inmediato. El valor predeterminado es true. Si es falso, `commit_transaction` realiza un sondeo y espera hasta que la transacción se haya confirmado. La cantidad de tiempo de espera se limita a un minuto mediante retroceso exponencial con un máximo de seis reintentos.

Devuelve un valor booleano para indicar si se realizó o no la confirmación.

cancel_transaction

`cancel_transaction(transaction_id)`

Intenta cancelar la transacción especificada. Devuelve una excepción de `TransactionCommittedException` si la transacción se había confirmado anteriormente. Internamente se llama CancelTransactionAPI Lake Formation.

- `transaction_id`: (cadena) la transacción que se cancelará.

Escritura

- getSink
- write_dynamic_frame_from_options
- write_from_options
- write_dynamic_frame_from_catalog

- write_data_frame_from_catalog
- write_dynamic_frame_from_jdbc_conf
- write_from_jdbc_conf

getSink

`getSink(connection_type, format = None, transformation_ctx = "", **options)`

Obtiene un objeto `DataSink` que se puede utilizar para escribir `DynamicFrames` en fuentes externas. Compruebe el valor de `format` de SparkSQL en primer lugar para asegurarse de que obtiene el receptor esperado.

- `connection_type`: tipo de conexión que se va a utilizar, como Amazon S3, Amazon Redshift y JDBC. Los valores válidos son `s3`, `mysql`, `postgresql`, `redshift`, `sqlserver`, `oracle`, `kinesis` y `kafka`.
- `format`: el formato de SparkSQL que se utilizará (opcional).
- `transformation_ctx`: contexto de transformación que se va a utilizar (opcional).
- `options`: conjunto de pares de nombre-valor que se utilizan para especificar las opciones de conexión. Algunos de valores posibles son:
  - `user` y `password`: para autorización
  - `url`: punto de conexión del almacén de datos
  - `dbtable`: nombre de la tabla de destino
  - `bulkSize`: grado de paralelismo para operaciones de inserción

Las opciones que puede especificar dependen del tipo de conexión. Consulte Tipos de conexión y opciones para ETL en AWS Glue para Spark para obtener más valores y ejemplos.

Ejemplo:

```
>>> data_sink = context.getSink("s3")
>>> data_sink.setFormat("json"),
>>> data_sink.writeFrame(myFrame)
```

write_dynamic_frame_from_options

`write_dynamic_frame_from_options(frame, connection_type, connection_options={}, format=None, format_options={}, transformation_ctx = "")`

Escribe y devuelve un `DynamicFrame` mediante la conexión y el formato especificados.

- `frame`: el `DynamicFrame` que se va a escribir.
- `connection_type`: el tipo de conexión, como Amazon S3, Amazon Redshift y JDBC. Los valores válidos son `s3`, `mysql`, `postgresql`, `redshift`, `sqlserver`, `oracle`, `kinesis` y `kafka`.
- `connection_options`: opciones de conexión, como la tabla de rutas y bases de datos (opcional). Para un `connection_type` de `s3`, se define una ruta de Amazon S3.

```
connection_options = {"path": "s3://aws-glue-target/temp"}
```

Para conexiones JDBC, deben definirse varias propiedades. Tenga en cuenta que el nombre de la base de datos debe ser parte de la URL. Opcionalmente, se puede incluir en las opciones de conexión.

> **Warning**
>
> No se recomienda almacenar las contraseñas en el script. Considere utilizarlos `boto3` para recuperarlos del AWS Secrets Manager catálogo de datos de AWS Glue.

```
connection_options = {"url": "jdbc-url/database", "user": "username",
 "password": passwordVariable,"dbtable": "table-name", "redshiftTmpDir": "s3-tempdir-path"}
```

La propiedad `dbtable` es el nombre de la tabla de JDBC. Para almacenes de datos de JDBC que admiten esquemas dentro de una base de datos, especifique `schema.table-name`. Si no se ha proporcionado un esquema, se usa el esquema "public" predeterminado.

Para obtener más información, consulte Tipos de conexión y opciones para ETL en AWS Glue para Spark.

- `format`— Una especificación de formato. Se utiliza con una conexión de Amazon S3 o AWS Glue que admite diversos formatos. Consulte en Opciones de formato de datos para las entradas y las salidas en AWS Glue para Spark los formatos que se admiten.
- `format_options`: opciones del formato especificado. Consulte en Opciones de formato de datos para las entradas y las salidas en AWS Glue para Spark los formatos que se admiten.
- `transformation_ctx`: contexto de transformación que se va a utilizar (opcional).

write_from_options

`write_from_options(frame_or_dfc, connection_type, connection_options={}, format={}, format_options={}, transformation_ctx = "")`

Escribe y devuelve un `DynamicFrame` o `DynamicFrameCollection` que se crea con la información sobre la conexión y el formato especificada.

- `frame_or_dfc`: el `DynamicFrame` o la `DynamicFrameCollection` que se van a escribir.
- `connection_type`: el tipo de conexión, como Amazon S3, Amazon Redshift y JDBC. Entre los valores válidos se incluyen: `s3`, `mysql`, `postgresql`, `redshift`, `sqlserver` y `oracle`.
- `connection_options`: opciones de conexión, como la tabla de rutas y bases de datos (opcional). Para un `connection_type` de `s3`, se define una ruta de Amazon S3.

```
connection_options = {"path": "s3://aws-glue-target/temp"}
```

Para conexiones JDBC, deben definirse varias propiedades. Tenga en cuenta que el nombre de la base de datos debe ser parte de la URL. Opcionalmente, se puede incluir en las opciones de conexión.

> **Warning**
>
> No se recomienda almacenar las contraseñas en el script. Considere utilizarlos `boto3` para recuperarlos del AWS Secrets Manager catálogo de datos de AWS Glue.

```
connection_options = {"url": "jdbc-url/database", "user": "username",
 "password": passwordVariable,"dbtable": "table-name", "redshiftTmpDir": "s3-tempdir-path"}
```

La propiedad `dbtable` es el nombre de la tabla de JDBC. Para almacenes de datos de JDBC que admiten esquemas dentro de una base de datos, especifique `schema.table-name`. Si no se ha proporcionado un esquema, se usa el esquema "public" predeterminado.

Para obtener más información, consulte Tipos de conexión y opciones para ETL en AWS Glue para Spark.

- `format`— Una especificación de formato. Se utiliza con una conexión de Amazon S3 o AWS Glue que admite diversos formatos. Consulte en Opciones de formato de datos para las entradas y las salidas en AWS Glue para Spark los formatos que se admiten.
- `format_options`: opciones del formato especificado. Consulte en Opciones de formato de datos para las entradas y las salidas en AWS Glue para Spark los formatos que se admiten.
- `transformation_ctx`: contexto de transformación que se va a utilizar (opcional).

write_dynamic_frame_from_catalog

`write_dynamic_frame_from_catalog(frame, database, table_name, redshift_tmp_dir, transformation_ctx = "", additional_options = {}, catalog_id = None)`

Escribe y devuelve un `DynamicFrame` que se crea mediante una base de datos y una tabla del Catálogo de datos.

- `frame`: el `DynamicFrame` que se va a escribir.
- `Database`: la base de datos del Catálogo de datos que contiene la tabla.
- `table_name`: el nombre de la tabla del Catálogo de datos asociada al destino.
- `redshift_tmp_dir`: directorio provisional de Amazon Redshift que se va a utilizar (opcional).
- `transformation_ctx`: contexto de transformación que se va a utilizar (opcional).
- `additional_options`: conjunto de pares nombre-valor opcionales.
- `catalog_id`: ID de catálogo (ID de cuenta) del Catálogo de datos al que se accede. Cuando el valor es Ninguno, se utiliza el ID de cuenta predeterminado del intermediario.

write_data_frame_from_catalog

`write_data_frame_from_catalog(frame, database, table_name, redshift_tmp_dir, transformation_ctx = "", additional_options = {}, catalog_id = None)`

Escribe y devuelve un `DataFrame` que se crea mediante una base de datos y una tabla del Catálogo de datos. Este método admite la escritura en formatos de lagos de datos (Hudi, Iceberg y Delta Lake). Para obtener más información, consulte Uso de marcos de lagos de datos con trabajos de ETL de AWS Glue.

- `frame`: el `DataFrame` que se va a escribir.
- `Database`: la base de datos del Catálogo de datos que contiene la tabla.
- `table_name`: nombre de la tabla del Catálogo de datos que está asociada al destino.
- `redshift_tmp_dir`: directorio provisional de Amazon Redshift que se va a utilizar (opcional).
- `transformation_ctx`: contexto de transformación que se va a utilizar (opcional).
- `additional_options`: conjunto de pares nombre-valor opcionales.
  - `useSparkDataSink`— Cuando se establece en true, AWS Glue debe usar la API nativa de Spark Data Sink para escribir en la tabla. Al activar esta opción, puedes añadir cualquier opción de fuente de datos de Spark `additional_options` según sea necesario. AWS Glue pasa estas opciones directamente a la grabadora Spark.
- `catalog_id`: el ID de catálogo (ID de cuenta) del Catálogo de datos al que se accede. Si no especifica un valor, se utiliza el ID de cuenta predeterminado del intermediario.

Limitaciones

Tenga en cuenta las siguientes limitaciones cuando utilice la opción `useSparkDataSink`:

- La opción `enableUpdateCatalog` no se admite cuando se utiliza la opción `useSparkDataSink`.

Ejemplo: escribir una tabla de Hudi con el escritor de orígenes de datos de Spark

```
hudi_options = {
    'useSparkDataSink': True,
    'hoodie.table.name': <table_name>,
    'hoodie.datasource.write.storage.type': 'COPY_ON_WRITE',
```

```
    'hoodie.datasource.write.recordkey.field': 'product_id',
    'hoodie.datasource.write.table.name': <table_name>,
    'hoodie.datasource.write.operation': 'upsert',
    'hoodie.datasource.write.precombine.field': 'updated_at',
    'hoodie.datasource.write.hive_style_partitioning': 'true',
    'hoodie.upsert.shuffle.parallelism': 2,
    'hoodie.insert.shuffle.parallelism': 2,
    'hoodie.datasource.hive_sync.enable': 'true',
    'hoodie.datasource.hive_sync.database': <database_name>,
    'hoodie.datasource.hive_sync.table': <table_name>,
    'hoodie.datasource.hive_sync.use_jdbc': 'false',
    'hoodie.datasource.hive_sync.mode': 'hms'}

glueContext.write_data_frame.from_catalog(
    frame = <df_product_inserts>,
    database = <database_name>,
    table_name = <table_name>,
    additional_options = hudi_options
)
```

write_dynamic_frame_from_jdbc_conf

`write_dynamic_frame_from_jdbc_conf(frame, catalog_connection, connection_options={}, redshift_tmp_dir = "", transformation_ctx = "", catalog_id = None)`

Escribe y devuelve un `DynamicFrame` mediante la información de la conexión JDBC especificada.

- `frame`: el `DynamicFrame` que se va a escribir.
- `catalog_connection`: conexión al catálogo que se va a utilizar.
- `connection_options`: opciones de conexión, como la tabla de rutas y bases de datos (opcional). Para obtener más información, consulte Tipos de conexión y opciones para ETL en AWS Glue para Spark.
- `redshift_tmp_dir`: directorio provisional de Amazon Redshift que se va a utilizar (opcional).
- `transformation_ctx`: contexto de transformación que se va a utilizar (opcional).
- `catalog_id`: ID de catálogo (ID de cuenta) del Catálogo de datos al que se accede. Cuando el valor es Ninguno, se utiliza el ID de cuenta predeterminado del intermediario.

write_from_jdbc_conf

`write_from_jdbc_conf(frame_or_dfc, catalog_connection, connection_options={}, redshift_tmp_dir = "", transformation_ctx = "", catalog_id = None)`

Escribe y devuelve un `DynamicFrame` o `DynamicFrameCollection` mediante la información de la conexión JDBC especificada.

- `frame_or_dfc`: el `DynamicFrame` o la `DynamicFrameCollection` que se van a escribir.
- `catalog_connection`: conexión al catálogo que se va a utilizar.
- `connection_options`: opciones de conexión, como la tabla de rutas y bases de datos (opcional). Para obtener más información, consulte Tipos de conexión y opciones para ETL en AWS Glue para Spark.
- `redshift_tmp_dir`: directorio provisional de Amazon Redshift que se va a utilizar (opcional).
- `transformation_ctx`: contexto de transformación que se va a utilizar (opcional).
- `catalog_id`: ID de catálogo (ID de cuenta) del Catálogo de datos al que se accede. Cuando el valor es Ninguno, se utiliza el ID de cuenta predeterminado del intermediario.

AWS Glue PySpark transforma la referencia

AWS Glue proporciona las siguientes transformaciones integradas que puede utilizar en las operaciones de PySpark ETL. Los datos pasan de una transformación a otra en una estructura de datos denominada a DynamicFrame, que es una extensión de un Apache Spark SQL`DataFrame`. `DynamicFrame` contiene sus datos y usted hace referencia a su esquema para procesar los datos.

La mayoría de estas transformaciones también existen como métodos de la clase `DynamicFrame`. Para obtener más información, consulte DynamicFrame transformaciones.

- Clase de base GlueTransform
- Clase ApplyMapping
- Clase DropFields
- Clase DropNullFields
- Clase ErrorsAsDynamicFrame
- Clase de EvaluateDataQuality

- Clase FillMissingValues
- Clase de filtro
- Clase FindIncrementalMatches
- Clase FindMatches
- Clase FlatMap
- Clase Join
- Clase Map
- Clase MapToCollection
- mergeDynamicFrame
- Clase Relationalize
- Clase RenameField
- Clase ResolveChoice
- Clase SelectFields
- Clase SelectFromCollection
- Clase simplify_ddb_json
- Clase Spigot
- Clase SplitFields
- Clase SplitRows
- Clase Unbox
- Clase UnnestFrame

Clase de base GlueTransform

La clase de base de la que heredan todas las clases de `awsglue.transforms`.

Todas las clases definen un método `__call__`. Pueden anular los métodos de la clase `GlueTransform` identificados en las siguientes secciones o bien, se invocan de forma predeterminada mediante el nombre de la clase.

Métodos

- apply(cls, *args, **kwargs)

- name(cls)
- describeArgs(cls)
- describeReturn(cls)
- describeTransform(cls)
- describeErrors(cls)
- describe(cls)

apply(cls, *args, **kwargs)

Aplica la transformación y, para ello, invoca a la clase de transformación y devuelve el resultado.

- `cls`: el objeto de la clase `self`.

name(cls)

Devuelve el nombre de la clase de transformación derivada.

- `cls`: el objeto de la clase `self`.

describeArgs(cls)

- `cls`: el objeto de la clase `self`.

Devuelve una lista de diccionarios, cada uno de los cuales se corresponde con un argumento designado, con el siguiente formato:

```
[
  {
    "name": "(name of argument)",
    "type": "(type of argument)",
    "description": "(description of argument)",
    "optional": "(Boolean, True if the argument is optional)",
    "defaultValue": "(Default value string, or None)(String; the default value, or
 None)"
  },
...
]
```

Genera una excepción `NotImplementedError` cuando se invoca en una transformación derivada en la que no se implementa.

describeReturn(cls)

- `cls`: el objeto de la clase `self`.

Devuelve un diccionario con información sobre el tipo de devolución, en el siguiente formato:

```
{
  "type": "(return type)",
  "description": "(description of output)"
}
```

Genera una excepción `NotImplementedError` cuando se invoca en una transformación derivada en la que no se implementa.

describeTransform(cls)

Devuelve una cadena que describe la transformación.

- `cls`: el objeto de la clase `self`.

Genera una excepción `NotImplementedError` cuando se invoca en una transformación derivada en la que no se implementa.

describeErrors(cls)

- `cls`: el objeto de la clase `self`.

Devuelve una lista de diccionarios donde cada uno de los cuales describe una posible excepción que genera esta transformación, con el siguiente formato:

```
[
  {
    "type": "(type of error)",
    "description": "(description of error)"
  },
...
```

```
]
```

describe(cls)

- `cls`: el objeto de la clase `self`.

Devuelve un objeto con el siguiente formato:

```
{
  "transform" : {
    "name" : cls.name( ),
    "args" : cls.describeArgs( ),
    "returns" : cls.describeReturn( ),
    "raises" : cls.describeErrors( ),
    "location" : "internal"
  }
}
```

Clase ApplyMapping

Aplica un mapeo en un `DynamicFrame`.

Ejemplo

Recomendamos que se utilice el método `DynamicFrame.apply_mapping()` para aplicar una asignación en un `DynamicFrame`. Para ver un ejemplo de código, consulte Ejemplo: uso de apply_mapping para cambiar el nombre de los campos y cambiar los tipos de campos.

Métodos

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- Describe

__call__(frame, mappings, transformation_ctx = "", info = "", stageThreshold = 0, totalThreshold = 0)

Aplica un mapeo declarativo a un `DynamicFrame` especificado.

- `frame`: `DynamicFrame` donde se aplica la asignación (obligatorio).
- `mappings`: una lista de tuplas de asignación (obligatorio). Cada una consta de (columna de origen, tipo de origen, columna de destino, tipo de destino).

  Si la columna de origen tiene un punto "`.`" en el nombre, se debe colocar acentos graves "`` ` ``" a su alrededor. Por ejemplo, para asignar `this.old.name` (cadena) a `thisNewName`, debe utilizar las siguientes tuplas:

  ```
  ("`this.old.name`", "string", "thisNewName", "string")
  ```

- `transformation_ctx`: cadena única que se utiliza para identificar la información del estado (opcional).
- `info`: cadena que está asociada con errores en la transformación (opcional).
- `stageThreshold`: cantidad máxima de errores que se puede producir en la transformación antes de que se determine que es errónea (opcional). El rol predeterminado es cero.
- `totalThreshold`: cantidad máxima de errores que se pueden producir en total antes de que se determine que el proceso es erróneo (opcional). El rol predeterminado es cero.

Devuelve solo los campos del `DynamicFrame` especificados en las tuplas de "asignación".

apply(cls, *args, **kwargs)

Heredado de `GlueTransform` apply.

name(cls)

Heredado de `GlueTransform` name.

describeArgs(cls)

Heredado de `GlueTransform` describeArgs.

describeReturn(cls)

Heredado de `GlueTransform` describeReturn.

describeTransform(cls)

Heredado de `GlueTransform` describeTransform.

describeErrors(cls)

Heredado de `GlueTransform` describeErrors.

describe(cls)

Heredado de `GlueTransform` describe.

Clase DropFields

Descarta campos en un `DynamicFrame`.

Ejemplo

Recomendamos que se utilice el método `DynamicFrame.drop_fields()` para descartar campos de un `DynamicFrame`. Para ver un ejemplo de código, consulte Ejemplo: uso de drop_fields para descartar campos de un `DynamicFrame`.

Métodos

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- Describe

__call__(frame, paths, transformation_ctx = "", info = "", stageThreshold = 0, totalThreshold = 0)

Descarta nodos en un `DynamicFrame`.

- `frame`: `DynamicFrame` donde se descartan los nodos (obligatorio).
- `paths`:lista de rutas completas a los nodos que se descartarán (obligatorio).

- `transformation_ctx`: cadena única que se utiliza para identificar la información del estado (opcional).
- `info`: cadena que está asociada a errores en la transformación (opcional).
- `stageThreshold`: cantidad máxima de errores que se puede producir en la transformación antes de que se determine que es errónea (opcional). El rol predeterminado es cero.
- `totalThreshold`: cantidad máxima de errores que se pueden producir en total antes de que se determine que el proceso es erróneo (opcional). El rol predeterminado es cero.

Devuelve un `DynamicFrame` nuevo sin los campos especificados.

apply(cls, *args, **kwargs)

Heredado de `GlueTransform` apply.

name(cls)

Heredado de `GlueTransform` name.

describeArgs(cls)

Heredado de `GlueTransform` describeArgs.

describeReturn(cls)

Heredado de `GlueTransform` describeReturn.

describeTransform(cls)

Heredado de `GlueTransform` describeTransform.

describeErrors(cls)

Heredado de `GlueTransform` describeErrors.

describe(cls)

Heredado de `GlueTransform` describe.

Clase DropNullFields

Descarta todos los campos con valores Null en un `DynamicFrame` cuyo tipo es `NullType`. Son campos incompletos o con valores Null en cada registro del conjunto de datos de `DynamicFrame`.

Ejemplo

En este ejemplo se utiliza `DropNullFields` para crear un `DynamicFrame` nuevo donde los campos de tipo `NullType` se han descartado. Para demostrar `DropNullFields`, agregamos una columna nueva denominada `empty_column` con tipo Null al conjunto de datos `persons` ya cargado.

> **Note**
>
> Para acceder al conjunto de datos utilizado en este ejemplo, consulte Código de ejemplo: unión de los datos y establecimiento de relaciones entre ellos y siga las instrucciones en Paso 1: Rastrear los datos del bucket de Amazon S3.

```
# Example: Use DropNullFields to create a new DynamicFrame without NullType fields

from pyspark.context import SparkContext
from awsglue.context import GlueContext
from pyspark.sql.functions import lit
from pyspark.sql.types import NullType
from awsglue.dynamicframe import DynamicFrame
from awsglue.transforms import DropNullFields

# Create GlueContext
sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

# Create DynamicFrame
persons = glueContext.create_dynamic_frame.from_catalog(
    database="legislators", table_name="persons_json"
)
print("Schema for the persons DynamicFrame:")
persons.printSchema()

# Add new column "empty_column" with NullType
persons_with_nulls = persons.toDF().withColumn("empty_column",
 lit(None).cast(NullType()))
persons_with_nulls_dyf = DynamicFrame.fromDF(persons_with_nulls, glueContext,
 "persons_with_nulls")
print("Schema for the persons_with_nulls_dyf DynamicFrame:")
persons_with_nulls_dyf.printSchema()
```

```
# Remove the NullType field
persons_no_nulls = DropNullFields.apply(persons_with_nulls_dyf)
print("Schema for the persons_no_nulls DynamicFrame:")
persons_no_nulls.printSchema()
```

Salida

```
Schema for the persons DynamicFrame:
root
|-- family_name: string
|-- name: string
|-- links: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- url: string
|-- gender: string
|-- image: string
|-- identifiers: array
|    |-- element: struct
|    |    |-- scheme: string
|    |    |-- identifier: string
|-- other_names: array
|    |-- element: struct
|    |    |-- lang: string
|    |    |-- note: string
|    |    |-- name: string
|-- sort_name: string
|-- images: array
|    |-- element: struct
|    |    |-- url: string
|-- given_name: string
|-- birth_date: string
|-- id: string
|-- contact_details: array
|    |-- element: struct
|    |    |-- type: string
|    |    |-- value: string
|-- death_date: string

Schema for the persons_with_nulls_dyf DynamicFrame:
root
|-- family_name: string
|-- name: string
```

```
|-- links: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- url: string
|-- gender: string
|-- image: string
|-- identifiers: array
|    |-- element: struct
|    |    |-- scheme: string
|    |    |-- identifier: string
|-- other_names: array
|    |-- element: struct
|    |    |-- lang: string
|    |    |-- note: string
|    |    |-- name: string
|-- sort_name: string
|-- images: array
|    |-- element: struct
|    |    |-- url: string
|-- given_name: string
|-- birth_date: string
|-- id: string
|-- contact_details: array
|    |-- element: struct
|    |    |-- type: string
|    |    |-- value: string
|-- death_date: string
|-- empty_column: null

null_fields ['empty_column']
Schema for the persons_no_nulls DynamicFrame:
root
|-- family_name: string
|-- name: string
|-- links: array
|    |-- element: struct
|    |    |-- note: string
|    |    |-- url: string
|-- gender: string
|-- image: string
|-- identifiers: array
|    |-- element: struct
|    |    |-- scheme: string
|    |    |-- identifier: string
```

```
|-- other_names: array
|    |-- element: struct
|    |    |-- lang: string
|    |    |-- note: string
|    |    |-- name: string
|-- sort_name: string
|-- images: array
|    |-- element: struct
|    |    |-- url: string
|-- given_name: string
|-- birth_date: string
|-- id: string
|-- contact_details: array
|    |-- element: struct
|    |    |-- type: string
|    |    |-- value: string
|-- death_date: string
```

Métodos

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- Describe

__call__(frame, transformation_ctx = "", info = "", stageThreshold = 0, totalThreshold = 0)

Descarta todos los campos con valores Null en un `DynamicFrame` cuyo tipo es `NullType`. Son campos incompletos o con valores Null en cada registro del conjunto de datos de `DynamicFrame`.

- `frame`: `DynamicFrame` donde se descartan los campos con valores Null (obligatorio).
- `transformation_ctx`: cadena única que se utiliza para identificar la información del estado (opcional).

- `info`: cadena que está asociada a errores en la transformación (opcional).
- `stageThreshold`: cantidad máxima de errores que se puede producir en la transformación antes de que se determine que es errónea (opcional). El rol predeterminado es cero.
- `totalThreshold`: cantidad máxima de errores que se pueden producir en total antes de que se determine que el proceso es erróneo (opcional). El rol predeterminado es cero.

Devuelve un `DynamicFrame` nuevo sin campos con valores Null.

apply(cls, *args, **kwargs)

- `cls`: cls

name(cls)

- `cls`: cls

describeArgs(cls)

- `cls`: cls

describeReturn(cls)

- `cls`: cls

describeTransform(cls)

- `cls`: cls

describeErrors(cls)

- `cls`: cls

describe(cls)

- `cls`: cls

Clase ErrorsAsDynamicFrame

Devuelve un `DynamicFrame` que contiene registros anidados de errores que ocurrieron mientras se creaba el `DynamicFrame` de origen.

Ejemplo

Le recomendamos que utilice el método `DynamicFrame.errorsAsDynamicFrame()` para recuperar y ver los registros de errores. Para ver un ejemplo de código, consulte Ejemplo: uso de errorsAsDynamicFrame para ver los registros de errores.

Métodos

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- Describe

__call__(frame)

Devuelve un `DynamicFrame` que contiene registros de errores anidados relacionados con el `DynamicFrame` de origen.

- `frame`: `DynamicFrame` de origen (obligatorio).

apply(cls, *args, **kwargs)

- `cls`: cls

name(cls)

- `cls`: cls

describeArgs(cls)

- `cls`: cls

describeReturn(cls)

- `cls`: cls

describeTransform(cls)

- `cls`: cls

describeErrors(cls)

- `cls`: cls

describe(cls)

- `cls`: cls

Clase de EvaluateDataQuality

Evalúa un conjunto de reglas de calidad de datos comparándolo con un `DynamicFrame` y devuelve un `DynamicFrame` nuevo con los resultados de la evaluación.

Ejemplo

En el siguiente código de ejemplo, se muestra cómo evaluar la calidad de los datos de un `DynamicFrame` y, a continuación, ver los resultados de la calidad de los datos.

```
from awsglue.transforms import *
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsgluedq.transforms import EvaluateDataQuality

#Create Glue context
sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)
```

```
# Define DynamicFrame
legislatorsAreas = glueContext.create_dynamic_frame.from_catalog(
    database="legislators", table_name="areas_json")

# Create data quality ruleset
ruleset = """Rules = [ColumnExists "id", IsComplete "id"]"""

# Evaluate data quality
dqResults = EvaluateDataQuality.apply(
    frame=legislatorsAreas,
    ruleset=ruleset,
    publishing_options={
        "dataQualityEvaluationContext": "legislatorsAreas",
        "enableDataQualityCloudWatchMetrics": True,
        "enableDataQualityResultsPublishing": True,
        "resultsS3Prefix": "DOC-EXAMPLE-BUCKET1",
    },
)


# Inspect data quality results
dqResults.printSchema()
dqResults.toDF().show()
```

Salida

```
root
|-- Rule: string
|-- Outcome: string
|-- FailureReason: string
|-- EvaluatedMetrics: map
|    |-- keyType: string
|    |-- valueType: double


+----------------------+-------+-------------+--------------------------------------+
|Rule                  |Outcome|FailureReason|EvaluatedMetrics                      |
+----------------------+-------+-------------+--------------------------------------+
|ColumnExists "id"     |Passed |null         |{}                                    |
|IsComplete "id"       |Passed |null         |{Column.first_name.Completeness -> 1.0}|
+----------------------+-------+-------------+--------------------------------------+
```

Métodos

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__(frame, ruleset, publishing_options = {})

- `frame`: el `DynamicFrame` del que desea evaluar la calidad de los datos.
- `ruleset`: un conjunto de reglas de lenguaje de definición de calidad de datos (DQDL) en formato de cadena. Para obtener más información sobre DQDL, consulte la guía de Referencia del lenguaje de definición de calidad de datos (DQDL).
- `publishing_options`: un diccionario que especifica las siguientes opciones para publicar los resultados y las métricas de la evaluación:
  - `dataQualityEvaluationContext`: una cadena que especifica el espacio de nombres en el que AWS Glue debe publicar las métricas de Amazon CloudWatch y los resultados de calidad de los datos. Las métricas agregadas aparecen en CloudWatch, mientras que los resultados completos aparecen en la interfaz de AWS Glue Studio.
    - Requerido: no
    - Valor predeterminado: `default_context`
  - `enableDataQualityCloudWatchMetrics`: especifica si los resultados de la evaluación de la calidad de los datos deben publicarse en CloudWatch. Especifique un espacio de nombres para las métricas mediante la opción `dataQualityEvaluationContext`.
    - Requerido: no
    - Valor predeterminado: False
  - `enableDataQualityResultsPublishing`: especifica si los resultados de calidad de los datos deben estar visibles en la pestaña Data Quality (Calidad de datos) de la interfaz de AWS Glue Studio.

  - Requerido: no
  - Valor predeterminado: true
- `resultsS3Prefix`: especifica la ubicación de Amazon S3 en la que AWS Glue puede escribir los resultados de la evaluación de la calidad de los datos.
  - Requerido: no
  - Valor predeterminado: "" (cadena vacía)

apply(cls, *args, **kwargs)

Heredado de `GlueTransform` apply.

name(cls)

Heredado de `GlueTransform` name.

describeArgs(cls)

Heredado de `GlueTransform` describeArgs.

describeReturn(cls)

Heredado de `GlueTransform` describeReturn.

describeTransform(cls)

Heredado de `GlueTransform` describeTransform.

describeErrors(cls)

Heredado de `GlueTransform` describeErrors.

describe(cls)

Heredado de `GlueTransform` describe.

Clase FillMissingValues

La clase `FillMissingValues` localiza valores nulos y cadenas vacías en un `DynamicFrame` especificado y utiliza métodos de machine learning, como regresión lineal y bosque aleatorio (random forest), para predecir los valores faltantes. El trabajo de ETL utiliza los valores del conjunto

de datos de entrada para formar al modelo de machine learning, que luego predice cuáles deben ser los valores que faltan.

> **Tip**
>
> Si utiliza conjuntos de datos progresivos, cada conjunto progresivo se utiliza como datos de formación para el modelo de machine learning, por lo que es posible que los resultados no sean tan precisos.

Para importar:

```
from awsglueml.transforms import FillMissingValues
```

Métodos

- Aplicar

aplicar (frame, missing_values_column, output_column ="", transformation_ctx ="", info ="", stageThreshold = 0, totalThreshold = 0)

Rellena los valores que faltan de un marco dinámico en una columna especificada y devuelve un nuevo marco con estimaciones en una nueva columna. Para filas sin valores faltantes, el valor de la columna especificada se duplica en la nueva columna.

- `frame`: el `DynamicFrame` en el que rellenar los valores que faltan. Obligatorio.
- `missing_values_column`: la columna que contiene valores faltantes (valores `null` y cadenas vacías). Obligatorio.
- `output_column`: el nombre de la nueva columna que contendrá valores estimados para todas las filas cuyo valor faltaba. Opcional; el valor predeterminado es el nombre de `missing_values_column` con el sufijo `"_filled"`.
- `transformation_ctx`: cadena única que se utiliza para identificar la información del estado (opcional).
- `info`: cadena que está asociada a errores en la transformación (opcional).
- `stageThreshold`: número máximo de errores que se pueden producir en la transformación antes de que se determine que es errónea (opcional, el valor predeterminado es cero).

- `totalThreshold`: número máximo de errores que se pueden producir en total antes de que se determine que el proceso es erróneo (opcional, el valor predeterminado es cero).

Devuelve un nuevo `DynamicFrame` con una columna adicional que contiene estimaciones para filas con valores faltantes y el valor actual para otras filas.

Clase de filtro

Crea un `DynamicFrame` nuevo que contiene registros de la entrada `DynamicFrame` que cumplen una función de predicado especificada.

Ejemplo

Recomendamos que se utilice el método `DynamicFrame.filter()` para filtrar registros en un `DynamicFrame`. Para ver un ejemplo de código, consulte Ejemplo: uso de filter para obtener una selección filtrada de campos.

Métodos

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__(frame, f, transformation_ctx="", info="", stageThreshold=0, totalThreshold=0))

Devuelve un `DynamicFrame` nuevo que se crea mediante la selección de registros de la entrada `DynamicFrame` que cumplen una función de predicado especificada.

- `frame`: el `DynamicFrame` de origen al que se va a aplicar la función de filtro especificada (obligatorio).
- `f`: la función de predicado que se aplicará a cada `DynamicRecord` en el `DynamicFrame`. La función debe tomar un `DynamicRecord` como argumento y devolver True (Verdadero) si el `DynamicRecord` cumple los requisitos del filtro o False (Falso) si no los cumple (obligatorio).

Un `DynamicRecord` representa un registro lógico en un `DynamicFrame`. Es similar a una fila en un `DataFrame` de Spark, salvo que es autodescriptivo y se puede utilizar para datos que no se ajustan a un esquema fijo.

- `transformation_ctx`: cadena única que se utiliza para identificar la información del estado (opcional).
- `info`: cadena que está asociada con errores en la transformación (opcional).
- `stageThreshold`: cantidad máxima de errores que se puede producir en la transformación antes de que se determine que es errónea (opcional). El rol predeterminado es cero.
- `totalThreshold`: cantidad máxima de errores que se pueden producir en total antes de que se determine que el proceso es erróneo (opcional). El rol predeterminado es cero.

apply(cls, *args, **kwargs)

Heredado de `GlueTransform` apply.

name(cls)

Heredado de `GlueTransform` name.

describeArgs(cls)

Heredado de `GlueTransform` describeArgs.

describeReturn(cls)

Heredado de `GlueTransform` describeReturn.

describeTransform(cls)

Heredado de `GlueTransform` describeTransform.

describeErrors(cls)

Heredado de `GlueTransform` describeErrors.

describe(cls)

Heredado de `GlueTransform` describe.

Clase FindIncrementalMatches

Identifica registros coincidentes en el `DynamicFrame` existente y progresivo, y crea un nuevo `DynamicFrame` con un identificador exclusivo asignado a cada grupo de registros coincidentes.

Para importar:

```
from awsglueml.transforms import FindIncrementalMatches
```

Métodos

- Aplicar

apply(existingFrame, incrementalFrame, transformId, transformation_ctx = "", info = "", stageThreshold = 0, totalThreshold = 0, enforcedMatches = none, computeMatchConfidenceScores = 0)

Identifica registros coincidentes en el `DynamicFrame` de entrada y crea un nuevo `DynamicFrame` con un identificador exclusivo asignado a cada grupo de registros coincidentes.

- `existingFrame`: el `DynamicFrame` existente y precoincidente para aplicar la transformación FindIncrementalMatches. Obligatorio.
- `incrementalFrame`: el `DynamicFrame` progresivo para aplicar la transformación FindIncrementalMatches para que coincida con el `existingFrame`. Obligatorio.
- `transformId`: el ID único asociado a la transformación FindIncrementalMatches para aplicarlo a los registros del `DynamicFrames`. Obligatorio.
- `transformation_ctx`: cadena única que se utiliza para identificar la información del estadísticas/estado. Opcional.
- `info`: cadena que está asociada a errores en la transformación. Opcional.
- `stageThreshold`: número máximo de errores que se pueden producir en la transformación antes de que se determine que es errónea. Opcional. El rol predeterminado es cero.
- `totalThreshold`: número máximo de errores que se pueden producir en total antes de que se determine que el proceso es erróneo. Opcional. El rol predeterminado es cero.
- `enforcedMatches`: el `DynamicFrame` utilizado para imponer coincidencias. Opcional. El valor predeterminado es Ninguno.

- `computeMatchConfidenceScores`: valor booleano que indica si debe calcularse una puntuación de confianza para cada grupo de registros coincidentes. Opcional. El valor predeterminado es false.

Devuelve un nuevo `DynamicFrame` con un identificador exclusivo asignado a cada grupo de registros coincidentes.

Clase FindMatches

Identifica registros coincidentes en el `DynamicFrame` de entrada y crea un nuevo `DynamicFrame` con un identificador exclusivo asignado a cada grupo de registros coincidentes.

Para importar:

```
from awsglueml.transforms import FindMatches
```

Métodos

- Aplicar

apply(frame, transformId, transformation_ctx = "", info = "", stageThreshold = 0, totalThreshold = 0, enforcedMatches = none, computeMatchConfidenceScores = 0)

Identifica registros coincidentes en el `DynamicFrame` de entrada y crea un nuevo `DynamicFrame` con un identificador exclusivo asignado a cada grupo de registros coincidentes.

- `frame`: el `DynamicFrame` para aplicar la transformación FindMatches. Obligatorio.
- `transformId`: el ID único asociado a la transformación FindMatches para aplicarlo a los registros del `DynamicFrame`. Obligatorio.
- `transformation_ctx`: cadena única que se utiliza para identificar la información del estadísticas/estado. Opcional.
- `info`: cadena que está asociada a errores en la transformación. Opcional.
- `stageThreshold`: número máximo de errores que se pueden producir en la transformación antes de que se determine que es errónea. Opcional. El rol predeterminado es cero.
- `totalThreshold`: número máximo de errores que se pueden producir en total antes de que se determine que el proceso es erróneo. Opcional. El rol predeterminado es cero.

- `enforcedMatches`: el `DynamicFrame` utilizado para imponer coincidencias. Opcional. El valor predeterminado es Ninguno.
- `computeMatchConfidenceScores`: valor booleano que indica si debe calcularse una puntuación de confianza para cada grupo de registros coincidentes. Opcional. El valor predeterminado es false.

Devuelve un nuevo `DynamicFrame` con un identificador exclusivo asignado a cada grupo de registros coincidentes.

Clase FlatMap

Aplica una transformación a cada `DynamicFrame` en una colección. Los resultados no se agrupan en un solo `DynamicFrame`, sino que se conservan como una colección.

Ejemplos de FlatMap

El siguiente fragmento de ejemplo muestra cómo utilizar la transformación `ResolveChoice` en un conjunto de marcos dinámicos cuando se aplica a un `FlatMap`. Los datos utilizados para la entrada se encuentran en el JSON ubicado en el marcador de posición de la dirección Amazon S3 `s3://bucket/path-for-data/sample.json` y contienen los siguientes datos.

Ejemplo de datos JSON

```
[{
    "firstname": "Arnav",
    "lastname": "Desai",
    "address": {
        "street": "6 Anyroad Avenue",
        "city": "London",
        "state": "England",
        "country": "UK"
    },
    "phone": 17235550101,
    "affiliations": [
        "General Anonymous Example Products",
        "Example Independent Research",
        "Government Department of Examples"
    ]
},
{
    "firstname": "Mary",
```

```
    "lastname": "Major",
    "address": {
        "street": "7821 Spot Place",
        "city": "Centerville",
        "state": "OK",
        "country": "US"
    },
    "phone": 19185550023,
    "affiliations": [
        "Example Dot Com",
        "Example Independent Research",
        "Example.io"
    ]
},
{
    "firstname": "Paulo",
    "lastname": "Santos",
    "address": {
        "street": "123 Maple Street",
        "city": "London",
        "state": "Ontario",
        "country": "CA"
    },
    "phone": 12175550181,
    "affiliations": [
        "General Anonymous Example Products",
        "Example Dot Com"
    ]
}]
```

Example Aplique ResolveChoice a una DynamicFrameCollection y muestre el resultado.

```
#Read DynamicFrame
datasource = glueContext.create_dynamic_frame_from_options("s3", connection_options =
 {"paths":["s3://bucket/path/to/file/mysamplejson.json"]}, format="json")
datasource.printSchema()
datasource.show()

## Split to create a DynamicFrameCollection
split_frame=datasource.split_fields(["firstname","lastname","address"],"personal_info","busines
split_frame.keys()
print("---")
```

```
## Use FlatMap to run ResolveChoice
kwargs = {"choice": "cast:string"}
flat = FlatMap.apply(split_frame, ResolveChoice, frame_name="frame",
 transformation_ctx='tcx', **kwargs)
flat.keys()

##Select one of the DynamicFrames
personal_info = flat.select("personal_info")
personal_info.printSchema()
personal_info.show()
print("---")

business_info = flat.select("business_info")
business_info.printSchema()
business_info.show()
```

> **Important**
>
> Al llamar a `FlatMap.apply`, el parámetro `frame_name` debe ser `"frame"`. Actualmente no se acepta ningún otro valor.

Ejemplo de resultado

```
root
|-- firstname: string
|-- lastname: string
|-- address: struct
|    |-- street: string
|    |-- city: string
|    |-- state: string
|    |-- country: string
|-- phone: long
|-- affiliations: array
|    |-- element: string
---
{
    "firstname": "Mary",
    "lastname": "Major",
    "address": {
        "street": "7821 Spot Place",
        "city": "Centerville",
```

```
        "state": "OK",
        "country": "US"
    },
    "phone": 19185550023,
    "affiliations": [
        "Example Dot Com",
        "Example Independent Research",
        "Example.io"
    ]
}

{
    "firstname": "Paulo",
    "lastname": "Santos",
    "address": {
        "street": "123 Maple Street",
        "city": "London",
        "state": "Ontario",
        "country": "CA"
    },
    "phone": 12175550181,
    "affiliations": [
        "General Anonymous Example Products",
        "Example Dot Com"
    ]
}
---
root
|-- firstname: string
|-- lastname: string
|-- address: struct
|    |-- street: string
|    |-- city: string
|    |-- state: string
|    |-- country: string

{
    "firstname": "Mary",
    "lastname": "Major",
    "address": {
        "street": "7821 Spot Place",
        "city": "Centerville",
        "state": "OK",
        "country": "US"
```

```
    }
}

{
    "firstname": "Paulo",
    "lastname": "Santos",
    "address": {
        "street": "123 Maple Street",
        "city": "London",
        "state": "Ontario",
        "country": "CA"
    }
}
---
root
|-- phone: long
|-- affiliations: array
|    |-- element: string

{
    "phone": 19185550023,
    "affiliations": [
        "Example Dot Com",
        "Example Independent Research",
        "Example.io"
    ]
}

{
    "phone": 12175550181,
    "affiliations": [
        "General Anonymous Example Products",
        "Example Dot Com"
    ]
}
```

Métodos

- __call__
- Aplicar
- Nombre
- describeArgs

- describeReturn
- describeTransform
- describeErrors
- Describe

__call__(dfc, BaseTransform, frame_name, transformation_ctx = "", **base_kwargs)

Aplica una transformación a cada `DynamicFrame` en una colección y aplana los resultados.

- `dfc`: el `DynamicFrameCollection` para aplanar (obligatorio).
- `BaseTransform`: una transformación derivada de `GlueTransform` que se aplica a cada miembro de la colección (obligatorio).
- `frame_name`: el nombre del argumento al que se pasan los elementos de la colección (obligatorio).
- `transformation_ctx`: cadena única que se utiliza para identificar la información del estado (opcional).
- `base_kwargs`: argumentos que se pasan a la transformación de base (obligatorio).

Devuelve un nuevo `DynamicFrameCollection` que se crea al aplicar la transformación a cada `DynamicFrame` en la `DynamicFrameCollection` de origen.

apply(cls, *args, **kwargs)

Heredado de `GlueTransform` apply.

name(cls)

Heredado de `GlueTransform` name.

describeArgs(cls)

Heredado de `GlueTransform` describeArgs.

describeReturn(cls)

Heredado de `GlueTransform` describeReturn.

describeTransform(cls)

Heredado de `GlueTransform` describeTransform.

describeErrors(cls)

Heredado de `GlueTransform` describeErrors.

describe(cls)

Heredado de `GlueTransform` describe.

Clase Join

Realiza una unión de igualdad en dos `DynamicFrames`.

Ejemplo

Le recomendamos que utilice el método `DynamicFrame.join()` para unir a `DynamicFrames`. Para ver un ejemplo de código, consulte Ejemplo: uso de join para combinar `DynamicFrames`.

Métodos

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- Describe

__call__(frame1, frame2, keys1, keys2, transformation_ctx = "")

Realiza una unión de igualdad en dos `DynamicFrames`.

- `frame1`: primer `DynamicFrame` que debe unirse (obligatorio).
- `frame2`: segundo `DynamicFrame` que debe unirse (obligatorio).
- `keys1`: claves que deben unirse en la primera trama (obligatorio).
- `keys2`: claves que deben unirse en la segunda trama (obligatorio).

- `transformation_ctx`: cadena única que se utiliza para identificar la información del estado (opcional).

Devuelve un `DynamicFrame` nuevo que se crea cuando se unen los dos `DynamicFrames`.

apply(cls, *args, **kwargs)

Heredado de `GlueTransform` apply.

name(cls)

Heredado de `GlueTransform` name.

describeArgs(cls)

Heredado de `GlueTransform` describeArgs.

describeReturn(cls)

Heredado de `GlueTransform` describeReturn.

describeTransform(cls)

Heredado de `GlueTransform` describeTransform.

describeErrors(cls)

Heredado de `GlueTransform` describeErrors.

describe(cls)

Heredado de `GlueTransform` describe.

Clase Map

Genera un nuevo `DynamicFrame` al aplicar una función a todos los registros de la entrada `DynamicFrame`.

Ejemplo

Recomendamos que se utilice el método `DynamicFrame.map()` para aplicar una función a todos los registros de un `DynamicFrame`. Para ver un ejemplo de código, consulte Ejemplo: uso de map para aplicar una función a cada registro en un `DynamicFrame`.

Métodos

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- Describe

__call__(frame, f, transformation_ctx="", info="", stageThreshold=0, totalThreshold=0)

Muestra un nuevo `DynamicFrame` que se obtiene al aplicar la función especificada a todos los `DynamicRecords` en el `DynamicFrame` original.

- `frame`: el `DynamicFrame` original al que se aplica la función de asignación (obligatorio).
- `f`: la función que se aplica a todos los `DynamicRecords` en el `DynamicFrame`. La función debe utilizar un `DynamicRecord` como argumento y devolver un `DynamicRecord` nuevo que se genera por la asignación (obligatorio).

  Un `DynamicRecord` representa un registro lógico en un `DynamicFrame`. Es similar a una fila en un `DataFrame` de Apache Spark, salvo que es autodescriptivo y se puede utilizar para datos que no se ajustan a un esquema fijo.
- `transformation_ctx`: cadena única que se utiliza para identificar la información del estado (opcional).
- `info`: cadena que está asociada a errores en la transformación (opcional).
- `stageThreshold`: cantidad máxima de errores que se puede producir en la transformación antes de que se determine que es errónea (opcional). El rol predeterminado es cero.
- `totalThreshold`: cantidad máxima de errores que se pueden producir en total antes de que se determine que el proceso es erróneo (opcional). El rol predeterminado es cero.

Muestra un nuevo `DynamicFrame` que se obtiene al aplicar la función especificada a todos los `DynamicRecords` en el `DynamicFrame` original.

apply(cls, *args, **kwargs)

Heredado de `GlueTransform` apply.

name(cls)

Heredado de `GlueTransform` name.

describeArgs(cls)

Heredado de `GlueTransform` describeArgs.

describeReturn(cls)

Heredado de `GlueTransform` describeReturn.

describeTransform(cls)

Heredado de `GlueTransform` describeTransform.

describeErrors(cls)

Heredado de `GlueTransform` describeErrors.

describe(cls)

Heredado de `GlueTransform` describe.

Clase MapToCollection

Aplica una transformación a cada `DynamicFrame` en la `DynamicFrameCollection` especificada.

Métodos

- __call__
- Aplicar
- Nombre
- describeArgs
- describeReturn
- describeTransform
- describeErrors

- Describe

__call__(dfc, BaseTransform, frame_name, transformation_ctx = "", **base_kwargs)

Aplica una función de transformación a cada `DynamicFrame` en la `DynamicFrameCollection` especificada.

- `dfc`: la `DynamicFrameCollection` sobre la que se aplica la función de transformación (obligatorio).
- `callable`: función de transformación que se puede invocar y que se aplica a cada miembro de la colección (obligatorio).
- `transformation_ctx`: cadena única que se utiliza para identificar la información del estado (opcional).

Devuelve un nuevo `DynamicFrameCollection` que se crea al aplicar la transformación a cada `DynamicFrame` en la `DynamicFrameCollection` de origen.

apply(cls, *args, **kwargs)

Heredado de `GlueTransform` apply

name(cls)

Heredado de `GlueTransform` name.

describeArgs(cls)

Heredado de `GlueTransform` describeArgs.

describeReturn(cls)

Heredado de `GlueTransform` describeReturn.

describeTransform(cls)

Heredado de `GlueTransform` describeTransform.

describeErrors(cls)

Heredado de `GlueTransform` describeErrors.

describe(cls)

Heredado de `GlueTransform` describe.

Clase Relationalize

Aplana un esquema anidado en un `DynamicFrame` y dinamiza columnas de matriz a partir del fotograma aplanado.

Ejemplo

Le recomendamos que utilice el método `DynamicFrame.relationalize()` para relacionalizar un `DynamicFrame`. Para ver un ejemplo de código, consulte Ejemplo: utilice relacionalizar para aplanar un esquema anidado en un `DynamicFrame`.

Métodos

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__(frame, staging_path=None, name='roottable', options=None, transformation_ctx = "", info = "", stageThreshold = 0, totalThreshold = 0)

Establece relaciones en un `DynamicFrame` y produce una lista de fotogramas que se genera al aplanar las columnas anidadas y dinamizar las columnas de matriz. La columna de matriz dinamizada puede unirse a la tabla raíz con la clave de combinación generada durante la fase de desanidado.

- `frame`: el `DynamicFrame` donde establecer relaciones (obligatorio).
- `staging_path`: ruta en la que el método puede almacenar las particiones de las tablas dinamizadas en formato CSV (opcional). Las tablas dinamizadas se leen desde esta ruta.
- `name`: el nombre de la tabla de raíz (opcional).

- `options`: diccionario de parámetros opcionales. En la actualidad no se utiliza.
- `transformation_ctx`: cadena única que se utiliza para identificar la información del estado (opcional).
- `info`: cadena que está asociada a errores en la transformación (opcional).
- `stageThreshold`: cantidad máxima de errores que se puede producir en la transformación antes de que se determine que es errónea (opcional). El rol predeterminado es cero.
- `totalThreshold`: cantidad máxima de errores que se pueden producir en total antes de que se determine que el proceso es erróneo (opcional). El rol predeterminado es cero.

apply(cls, *args, **kwargs)

Heredado de `GlueTransform` apply.

name(cls)

Heredado de `GlueTransform` name.

describeArgs(cls)

Heredado de `GlueTransform` describeArgs.

describeReturn(cls)

Heredado de `GlueTransform` describeReturn.

describeTransform(cls)

Heredado de `GlueTransform` describeTransform.

describeErrors(cls)

Heredado de `GlueTransform` describeErrors.

describe(cls)

Heredado de `GlueTransform` describe.

Clase RenameField

Cambia el nombre de un nodo dentro de un `DynamicFrame`.

Ejemplo

Recomendamos que se utilice el método `DynamicFrame.rename_field()` para renombrar registros en un `DynamicFrame`. Para ver un ejemplo de código, consulte Ejemplo: utilice rename_field para cambiar el nombre de los campos de un `DynamicFrame`.

Métodos

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__(frame, old_name, new_name, transformation_ctx = "", info = "", stageThreshold = 0, totalThreshold = 0)

Cambia el nombre de un nodo dentro de un `DynamicFrame`.

- `frame`: `DynamicFrame` donde se cambia el nombre de un nodo (obligatorio).
- `old_name`: ruta completa al nodo cuyo nombre se quiere cambiar (obligatorio).

  Si el nombre anterior contiene puntos, RenameField no funcionará a menos que lo ponga entre acentos graves (``). Por ejemplo, para reemplazar `this.old.name` por `thisNewName`, llame a RenameField como se indica a continuación:

  ```
  newDyF = RenameField(oldDyF, "`this.old.name`", "thisNewName")
  ```

- `new_name`: nuevo nombre, incluida la ruta completa (obligatorio).
- `transformation_ctx`: cadena única que se utiliza para identificar la información del estado (opcional).
- `info`: cadena que está asociada a errores en la transformación (opcional).
- `stageThreshold`: cantidad máxima de errores que se puede producir en la transformación antes de que se determine que es errónea (opcional). El rol predeterminado es cero.

- `totalThreshold`: cantidad máxima de errores que se pueden producir en total antes de que se determine que el proceso es erróneo (opcional). El rol predeterminado es cero.

apply(cls, *args, **kwargs)

Heredado de `GlueTransform` apply.

name(cls)

Heredado de `GlueTransform` name.

describeArgs(cls)

Heredado de `GlueTransform` describeArgs.

describeReturn(cls)

Heredado de `GlueTransform` describeReturn.

describeTransform(cls)

Heredado de `GlueTransform` describeTransform.

describeErrors(cls)

Heredado de `GlueTransform` describeErrors.

describe(cls)

Heredado de `GlueTransform` describe.

Clase ResolveChoice

Resuelve un tipo de elección dentro de un elemento `DynamicFrame`.

Ejemplo

Se recomienda utilizar el método `DynamicFrame.resolveChoice()` para gestionar los campos que contienen varios tipos en un `DynamicFrame`. Para ver un ejemplo de código, consulte Ejemplo: utilice resolveChoice para gestionar una columna que contiene varios tipos.

Métodos

- __call__
- apply

- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__(frame, specs = none, choice = "", transformation_ctx = "", info = "", stageThreshold = 0, totalThreshold = 0)

Proporciona información para resolver tipos ambiguos dentro de un elemento `DynamicFrame`. Devuelve el elemento `DynamicFrame` resultante.

- `frame`: el elemento `DynamicFrame` en el que debe resolverse el tipo de elección (obligatorio).
- `specs`: lista de ambigüedades concretas que deben resolverse; cada una en forma de tupla: `(path, action)`. El valor de `path` identifica un elemento ambiguo concreto, mientras que el valor de `action` identifica la resolución correspondiente.

  Solo se puede utilizar el parámetro `spec` y el parámetro `choice`. Si el parámetro `spec` no es `None`, el parámetro `choice` tiene que ser una cadena vacía. Y viceversa, si el parámetro `choice` no es una cadena vacía, el parámetro `spec` tiene que ser `None`. Si no se proporciona ninguno de estos parámetros, AWS Glue intenta analizar el esquema y utilizarlo para resolver ambigüedades.

  Puede especificar una de las cuatro estrategias de resolución en la parte `action` de una tupla `specs`:

  - `cast`: le permite especificar el tipo al que se realizará una conversión (por ejemplo, `cast:int`).
  - `make_cols`: resuelve una posible ambigüedad al aplanar los datos. Por ejemplo, si `columnA` puede ser tanto un `int` como una `string`, la resolución consiste en producir dos columnas llamadas `columnA_int` y `columnA_string` en el elemento `DynamicFrame` resultante.
  - `make_struct`: resuelve una posible ambigüedad con una estructura para representar los datos. Por ejemplo, si los datos de una columna pueden ser tanto un `int` como una `string`, si usa la acción `make_struct` generará una columna de estructuras en el elemento `DynamicFrame` resultante, en la que cada estructura contendrá un `int` y una `string`.
  - `project`: resuelve una posible ambigüedad conservando solo los valores de un tipo especificado en el `DynamicFrame` resultante. Por ejemplo, si los datos de una columna

ChoiceType pueden ser un int o una string, al especificar una acción project:string se descartan los valores del DynamicFrame que no son del tipo string.

Si en path se identifica una matriz, incluya corchetes vacíos después del nombre de la matriz para evitar ambigüedades. Por ejemplo, suponga que está trabajando con datos estructurados tal y como se indica a continuación:

```
"myList": [
  { "price": 100.00 },
  { "price": "$100.00" }
]
```

Puede seleccionar la versión numérica en vez de la de cadena del precio si configura path en "myList[].price" y action en "cast:double".

- choice: la acción de resolución predeterminada si el parámetro specs es None. Si el parámetro specs no es None, este parámetro no tiene que establecerse en nada, sino que debe dejarse como una cadena vacía.

  Además de las acciones descritas anteriormente para specs, este argumento también admite la siguiente acción:

  - MATCH_CATALOG: intenta convertir cada ChoiceType al tipo correspondiente en la tabla de Data Catalog especificada.
- database: la base de datos del catálogo de datos de AWS Glue que se utilizará con la elección MATCH_CATALOG (obligatoria para MATCH_CATALOG).
- table_name: el nombre de la tabla del catálogo de datos de AWS Glue que se utilizará con la elección MATCH_CATALOG (obligatoria para MATCH_CATALOG).
- transformation_ctx: cadena única que se utiliza para identificar la información del estado (opcional).
- info: cadena que está asociada a errores en la transformación (opcional).
- stageThreshold: cantidad máxima de errores que se puede producir en la transformación antes de que se determine que es errónea (opcional). El rol predeterminado es cero.
- totalThreshold: cantidad máxima de errores que se pueden producir en total antes de que se determine que el proceso es erróneo (opcional). El rol predeterminado es cero.

apply(cls, *args, **kwargs)

Heredado de `GlueTransform` apply.

name(cls)

Heredado de `GlueTransform` name.

describeArgs(cls)

Heredado de `GlueTransform` describeArgs.

describeReturn(cls)

Heredado de `GlueTransform` describeReturn.

describeTransform(cls)

Heredado de `GlueTransform` describeTransform.

describeErrors(cls)

Heredado de `GlueTransform` describeErrors.

describe(cls)

Heredado de `GlueTransform` describe.

Clase SelectFields

La clase `SelectFields` crea un `DynamicFrame` nuevo a partir de un `DynamicFrame` existente, y conserva solo los campos que se especifican. `SelectFields` brinda una funcionalidad similar a la de una instrucción `SELECT` en SQL.

Ejemplo

Recomendamos que se utilice el método `DynamicFrame.select_fields()` para seleccionar campos de un `DynamicFrame`. Para ver un ejemplo de código, consulte Ejemplo: uso de select_fields para crear un `DynamicFrame` nuevo con los campos elegidos.

Métodos

- __call__
- apply

- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- Describe

__call__(frame, paths, transformation_ctx = "", info = "", stageThreshold = 0, totalThreshold = 0)

Obtiene campos (nodos) en un `DynamicFrame`.

- `frame`: `DynamicFrame` donde se seleccionan los campos (obligatorio).
- `paths`: una lista de rutas completas a los campos que se van a seleccionar (obligatorio).
- `transformation_ctx`: cadena única que se utiliza para identificar la información del estado (opcional).
- `info`: cadena que está asociada con errores en la transformación (opcional).
- `stageThreshold`: cantidad máxima de errores que se puede producir en la transformación antes de que se determine que es errónea (opcional). El rol predeterminado es cero.
- `totalThreshold`: cantidad máxima de errores que se pueden producir en total antes de que se determine que el proceso es erróneo (opcional). El rol predeterminado es cero.

Devuelve un `DynamicFrame` nuevo que contiene solo los campos especificados.

apply(cls, *args, **kwargs)

Heredado de `GlueTransform` apply.

name(cls)

Heredado de `GlueTransform` name.

describeArgs(cls)

Heredado de `GlueTransform` describeArgs.

describeReturn(cls)

Heredado de `GlueTransform` describeReturn.

describeTransform(cls)

Heredado de `GlueTransform` describeTransform.

describeErrors(cls)

Heredado de `GlueTransform` describeErrors.

describe(cls)

Heredado de `GlueTransform` describe.

Clase SelectFromCollection

Selecciona un `DynamicFrame` en un `DynamicFrameCollection`.

Ejemplo

En este ejemplo se utiliza `SelectFromCollection` para seleccionar un `DynamicFrame` de una`DynamicFrameCollection`.

Conjunto de datos de ejemplo

En el ejemplo se seleccionan dos `DynamicFrames` de un `DynamicFrameCollection` llamado `split_rows_collection`. A continuación se ofrece una lista de las claves de `split_rows_collection`.

```
dict_keys(['high', 'low'])
```

Código de ejemplo

```
# Example: Use SelectFromCollection to select
# DynamicFrames from a DynamicFrameCollection

from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.transforms import SelectFromCollection

# Create GlueContext
sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)
```

```
# Select frames and inspect entries
frame_low = SelectFromCollection.apply(dfc=split_rows_collection, key="low")
frame_low.toDF().show()

frame_high = SelectFromCollection.apply(dfc=split_rows_collection, key="high")
frame_high.toDF().show()
```

Salida

```
+---+-----+-----------------------+-----------------------+
| id|index|contact_details.val.type|contact_details.val.value|
+---+-----+-----------------------+-----------------------+
|  1|    0|                    fax|           202-225-3307|
|  1|    1|                  phone|           202-225-5731|
|  2|    0|                    fax|           202-225-3307|
|  2|    1|                  phone|           202-225-5731|
|  3|    0|                    fax|           202-225-3307|
|  3|    1|                  phone|           202-225-5731|
|  4|    0|                    fax|           202-225-3307|
|  4|    1|                  phone|           202-225-5731|
|  5|    0|                    fax|           202-225-3307|
|  5|    1|                  phone|           202-225-5731|
|  6|    0|                    fax|           202-225-3307|
|  6|    1|                  phone|           202-225-5731|
|  7|    0|                    fax|           202-225-3307|
|  7|    1|                  phone|           202-225-5731|
|  8|    0|                    fax|           202-225-3307|
|  8|    1|                  phone|           202-225-5731|
|  9|    0|                    fax|           202-225-3307|
|  9|    1|                  phone|           202-225-5731|
| 10|    0|                    fax|           202-225-6328|
| 10|    1|                  phone|           202-225-4576|
+---+-----+-----------------------+-----------------------+
only showing top 20 rows

+---+-----+-----------------------+-----------------------+
| id|index|contact_details.val.type|contact_details.val.value|
+---+-----+-----------------------+-----------------------+
| 11|    0|                    fax|           202-225-6328|
| 11|    1|                  phone|           202-225-4576|
| 11|    2|                twitter|          RepTrentFranks|
| 12|    0|                    fax|           202-225-6328|
| 12|    1|                  phone|           202-225-4576|
```

```
| 12|    2|                twitter|           RepTrentFranks|
| 13|    0|                    fax|             202-225-6328|
| 13|    1|                  phone|             202-225-4576|
| 13|    2|                twitter|           RepTrentFranks|
| 14|    0|                    fax|             202-225-6328|
| 14|    1|                  phone|             202-225-4576|
| 14|    2|                twitter|           RepTrentFranks|
| 15|    0|                    fax|             202-225-6328|
| 15|    1|                  phone|             202-225-4576|
| 15|    2|                twitter|           RepTrentFranks|
| 16|    0|                    fax|             202-225-6328|
| 16|    1|                  phone|             202-225-4576|
| 16|    2|                twitter|           RepTrentFranks|
| 17|    0|                    fax|             202-225-6328|
| 17|    1|                  phone|             202-225-4576|
+---+-----+-----------------------+-------------------------+
only showing top 20 rows
```

Métodos

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__(dfc, key, transformation_ctx = "")

Obtiene un `DynamicFrame` de una `DynamicFrameCollection`.

- `dfc`: la `DynamicFrameCollection` de la que el `DynamicFrame` debe seleccionarse (obligatorio).
- `key`: la clave del `DynamicFrame` que se debe seleccionar (obligatorio).

- `transformation_ctx`: cadena única que se utiliza para identificar la información del estado (opcional).

apply(cls, *args, **kwargs)

Heredado de `GlueTransform` apply.

name(cls)

Heredado de `GlueTransform` name.

describeArgs(cls)

Heredado de `GlueTransform` describeArgs.

describeReturn(cls)

Heredado de `GlueTransform` describeReturn.

describeTransform(cls)

Heredado de `GlueTransform` describeTransform.

describeErrors(cls)

Heredado de `GlueTransform` describeErrors.

describe(cls)

Heredado de `GlueTransform` describe.

Clase simplify_ddb_json

Simplifica las columnas anidadas en un `DynamicFrame` que se encuentran de manera específica en la estructura de JSON de DynamoDB y devuelve un nuevo `DynamicFrame` simplificado.

Ejemplo

Se recomienda utilizar el método `DynamicFrame.simplify_ddb_json()` para simplificar las columnas anidadas en un `DynamicFrame` que se encuentran de manera específica en la

estructura de JSON de DynamoDB. Para ver un ejemplo de código, consulte Por ejemplo: utilice simplify_ddb_json para invocar una simplificación JSON de DynamoDB..

Clase Spigot

Escribe registros de ejemplo en un destino especificado para ayudarlo a verificar las transformaciones hechas por su trabajo de AWS Glue.

Ejemplo

Se recomienda utilizar el método `DynamicFrame.spigot()` para escribir un subconjunto de registros desde un `DynamicFrame` a un destino especificado. Para ver un ejemplo de código, consulte Ejemplo: utilice spigot para escribir campos de ejemplo desde un `DynamicFrame` en Amazon S3.

Métodos

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__(frame, path, options, transformation_ctx = "")

Escribe registros de muestra en un destino especificado durante una transformación.

- `frame`: `DynamicFrame` en el que se ejecuta spigot (obligatorio).
- `path`: ruta al destino en el que se escribirá (obligatorio).
- `options`: pares clave-valor JSON que especifican opciones (opcional). La opción `"topk"` especifica que deben escribirse los primeros registros k. La opción `"prob"` especifica la probabilidad (como decimal) de seleccionar cualquier registro. Esto se utiliza para seleccionar los registros que se desean escribir.

- `transformation_ctx`: cadena única que se utiliza para identificar la información del estado (opcional).

apply(cls, *args, **kwargs)

Heredado de `GlueTransform` apply

name(cls)

Heredado de `GlueTransform` name

describeArgs(cls)

Heredado de `GlueTransform` describeArgs

describeReturn(cls)

Heredado de `GlueTransform` describeReturn

describeTransform(cls)

Heredado de `GlueTransform` describeTransform

describeErrors(cls)

Heredado de `GlueTransform` describeErrors

describe(cls)

Heredado de `GlueTransform` describe

Clase SplitFields

Divide un elemento `DynamicFrame` en dos elementos nuevos según los campos especificados.

Ejemplo

Recomendamos que se utilice el método `DynamicFrame.split_fields()` para dividir campos en un `DynamicFrame`. Para ver un ejemplo de código, consulte Ejemplo: utilice split_fields para dividir los campos seleccionados en un campo `DynamicFrame` independiente.

Métodos

- __call__

- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__(frame, paths, name1 = none, name2 = none, transformation_ctx = "", info = "", stageThreshold = 0, totalThreshold = 0)

Divide uno o más campos de un elemento `DynamicFrame` en un elemento `DynamicFrame` nuevo y crea otro elemento `DynamicFrame` nuevo que contiene los campos restantes.

- `frame`: `DynamicFrame` de origen que se va a dividir en dos nuevos (obligatorio).
- `paths`: lista de rutas completas a los campos que se van a dividir (obligatorio).
- `name1`: nombre que se asigna al `DynamicFrame` que contendrá los campos que se van a dividir (opcional). Si no se da ningún nombre, se usa el nombre de la trama de origen con un "1" anexado.
- `name2`: nombre que se asigna al `DynamicFrame` que contendrá los campos que quedan tras haber dividido los campos que se especificaron (opcional). Si no se da ningún nombre, se usa el nombre de la trama de origen con un "2" anexado.
- `transformation_ctx`: cadena única que se utiliza para identificar la información del estado (opcional).
- `info`: cadena que está asociada a errores en la transformación (opcional).
- `stageThreshold`: cantidad máxima de errores que se puede producir en la transformación antes de que se determine que es errónea (opcional). El rol predeterminado es cero.
- `totalThreshold`: cantidad máxima de errores que se pueden producir en total antes de que se determine que el proceso es erróneo (opcional). El rol predeterminado es cero.

apply(cls, *args, **kwargs)

Heredado de `GlueTransform` apply.

name(cls)

Heredado de `GlueTransform` name.

describeArgs(cls)

Heredado de `GlueTransform` describeArgs.

describeReturn(cls)

Heredado de `GlueTransform` describeReturn.

describeTransform(cls)

Heredado de `GlueTransform` describeTransform.

describeErrors(cls)

Heredado de `GlueTransform` describeErrors.

describe(cls)

Heredado de `GlueTransform` describe.

Clase SplitRows

Crea una `DynamicFrameCollection` que contiene dos `DynamicFrames`. Un `DynamicFrame` contiene solo las filas especificadas para dividirlas y el otro las filas restantes.

Ejemplo

Le recomendamos que utilice el método `DynamicFrame.split_rows()` para dividir filas en un `DynamicFrame`. Para ver un ejemplo de código, consulte Ejemplo: utilice split_rows para dividir filas en un `DynamicFrame`.

Métodos

- __call__
- apply
- name
- describeArgs

- describeReturn
- describeTransform
- describeErrors
- describe

__call__(frame, comparison_dict, name1="frame1", name2="frame2", transformation_ctx = "", info = none, stageThreshold = 0, totalThreshold = 0)

Divide una o más filas de un `DynamicFrame` en un nuevo `DynamicFrame`.

- `frame`: `DynamicFrame` de origen que se va a dividir en dos nuevos (obligatorio).
- `comparison_dict`: diccionario donde la clave es la ruta completa a una columna y el valor es otro diccionario que mapea comparadores con los valores con los que se comparan los valores de columna. Por ejemplo, `{"age": {">": 10, "<": 20}}` separa las filas en las que el valor "edad" se sitúa entre 10 y 20, exclusivamente, de las filas donde "edad" no se inscribe en ese intervalo (obligatorio).
- `name1`: el nombre que se asigna al `DynamicFrame` que contendrá las filas que se van a dividir (opcional).
- `name2`: el nombre que se asigna al `DynamicFrame` que contendrá las filas que quedan tras haber dividido las filas que se especificaron (opcional).
- `transformation_ctx`: cadena única que se utiliza para identificar la información del estado (opcional).
- `info`: cadena que está asociada a errores en la transformación (opcional).
- `stageThreshold`: cantidad máxima de errores que se puede producir en la transformación antes de que se determine que es errónea (opcional). El rol predeterminado es cero.
- `totalThreshold`: cantidad máxima de errores que se pueden producir en total antes de que se determine que el proceso es erróneo (opcional). El rol predeterminado es cero.

apply(cls, *args, **kwargs)

Heredado de `GlueTransform` apply.

name(cls)

Heredado de `GlueTransform` name.

describeArgs(cls)

Heredado de `GlueTransform` describeArgs.

describeReturn(cls)

Heredado de `GlueTransform` describeReturn.

describeTransform(cls)

Heredado de `GlueTransform` describeTransform.

describeErrors(cls)

Heredado de `GlueTransform` describeErrors.

describe(cls)

Heredado de `GlueTransform` describe.

Clase Unbox

Conversión unbox (nuevo formato) de un campo de cadena en un `DynamicFrame`.

Ejemplo

Recomendamos que se utilice el método `DynamicFrame.unbox()` para la conversión unbox en un `DynamicFrame`. Para ver un ejemplo de código, consulte Ejemplo: utilice la conversión unboxing para convertir un campo de cadena en una estructura.

Métodos

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors

- describe

__call__(frame, path, format, transformation_ctx = "", info="", stageThreshold=0, totalThreshold=0, **options)

Conversión unbox de un campo de cadena en un elemento `DynamicFrame`.

- `frame`: el `DynamicFrame` en el que se realizará la conversión unbox de un campo. (Obligatorio).
- `path`: ruta completa al elemento `StringNode` cuya conversión unbox debe realizarse (obligatorio).
- `format`: una especificación de formato (opcional). Se utiliza para una conexión de Amazon S3 o de AWS Glue que admita diversos formatos. Consulte en Opciones de formato de datos para las entradas y las salidas en AWS Glue para Spark sobre los formatos que se admiten.
- `transformation_ctx`: cadena única que se utiliza para identificar la información del estado (opcional).
- `info`: cadena que está asociada a errores en la transformación (opcional).
- `stageThreshold`: cantidad máxima de errores que se puede producir en la transformación antes de que se determine que es errónea (opcional). El rol predeterminado es cero.
- `totalThreshold`: cantidad máxima de errores que se pueden producir en total antes de que se determine que el proceso es erróneo (opcional). El rol predeterminado es cero.
- `separator`: token de separador (opcional).
- `escaper`: token de escape (opcional).
- `skipFirst`: `True` si la primera línea de datos debe omitirse o `False` si no debe omitirse (opcional).
- withSchema: cadena que contiene esquemas para los datos cuya conversión unbox debe realizarse (opcional). Debe crear siempre utilizando `StructType.json`.
- `withHeader`: `True` si los datos cuya conversión unbox se realiza contienen un encabezado o `False` en caso contrario (opcional).

apply(cls, *args, **kwargs)

Heredado de `GlueTransform` apply.

name(cls)

Heredado de `GlueTransform` name.

describeArgs(cls)

Heredado de `GlueTransform` describeArgs.

describeReturn(cls)

Heredado de `GlueTransform` describeReturn.

describeTransform(cls)

Heredado de `GlueTransform` describeTransform.

describeErrors(cls)

Heredado de `GlueTransform` describeErrors.

describe(cls)

Heredado de `GlueTransform` describe.

Clase UnnestFrame

Aplana `DynamicFrame` y los objetos anidados hasta convertirlos en elementos de nivel superior y genera claves de combinación para los objetos de matriz.

Ejemplo

Le recomendamos que utilice el método `DynamicFrame.unnest()` para aplanar estructuras anidadas en un `DynamicFrame`. Para ver un ejemplo de código, consulte Ejemplo: utilice desanidar para convertir los campos anidados en campos de nivel superior.

Métodos

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors

- describe

__call__(frame, transformation_ctx = "", info="", stageThreshold=0, totalThreshold=0)

Aplana `DynamicFrame` y los objetos anidados hasta convertirlos en elementos de nivel superior y genera claves de combinación para los objetos de matriz.

- `frame`: `DynamicFrame` que se va a aplanar (obligatorio).
- `transformation_ctx`: cadena única que se utiliza para identificar la información del estado (opcional).
- `info`: cadena que está asociada a errores en la transformación (opcional).
- `stageThreshold`: cantidad máxima de errores que se puede producir en la transformación antes de que se determine que es errónea (opcional). El rol predeterminado es cero.
- `totalThreshold`: cantidad máxima de errores que se pueden producir en total antes de que se determine que el proceso es erróneo (opcional). El rol predeterminado es cero.

apply(cls, *args, **kwargs)

Heredado de `GlueTransform` apply.

name(cls)

Heredado de `GlueTransform` name.

describeArgs(cls)

Heredado de `GlueTransform` describeArgs.

describeReturn(cls)

Heredado de `GlueTransform` describeReturn.

describeTransform(cls)

Heredado de `GlueTransform` describeTransform.

describeErrors(cls)

Heredado de `GlueTransform` describeErrors.

describe(cls)

Heredado de `GlueTransform` describe.

FlagDuplicatesInColumn clase

La `FlagDuplicatesInColumn` transformación devuelve una nueva columna con un valor específico en cada fila que indica si el valor de la columna de origen de la fila coincide con un valor de una fila anterior de la columna de origen. Cuando se encuentran coincidencias, se marcan como duplicadas. La aparición inicial no está marcada porque no coincide con una fila anterior.

Ejemplo

```
from pyspark.context import SparkContext
from pyspark.sql import SparkSession
from awsgluedi.transforms import *

sc = SparkContext()
spark = SparkSession(sc)

datasource1 = spark.read.json("s3://${BUCKET}/json/zips/raw/data")

try:
    df_output = column.FlagDuplicatesInColumn.apply(
        data_frame=datasource1,
        spark_context=sc,
        source_column="city",
        target_column="flag_col",
        true_string="True",
        false_string="False"
    )
except:
    print("Unexpected Error happened ")
    raise
```

Salida

La `FlagDuplicatesInColumn` transformación añadirá una nueva columna, `flag_col`, a `df_output`. DataFrame Esta columna contendrá un valor de cadena que indicará si la fila correspondiente tiene un valor duplicado en la columna `ciudad` o no. Si una fila tiene un valor de `ciudad` duplicado, `flag_col` contendrá el valor `true_string` «True». Si una fila tiene un valor de `ciudad` único, `flag_col` contendrá el valor `false_string` «False».

La opción `df_output` resultante DataFrame contendrá todas las columnas de la `datasource1` original, además de la columna adicional `flag_col` que indica valores de `ciudad` duplicados. DataFrame

Métodos

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__ (spark_context, data_frame, source_column, target_column, true_string=DEFAULT_TRUE_STRING, FALSE_STRING=DEFAULT_FALSE_STRING)

La `FlagDuplicatesInColumn` transformación devuelve una nueva columna con un valor específico en cada fila que indica si el valor de la columna de origen de la fila coincide con un valor de una fila anterior de la columna de origen. Cuando se encuentran coincidencias, se marcan como duplicadas. La aparición inicial no está marcada porque no coincide con una fila anterior.

- `source_column`— Nombre de la columna de origen.
- `target_column`— Nombre de la columna de destino.
- `true_string`— Cadena que se insertará en la columna de destino cuando el valor de una columna de origen duplique un valor anterior de esa columna.
- `false_string`— Cadena que se insertará en la columna de destino cuando el valor de la columna de origen sea distinto de los valores anteriores de esa columna.

apply(cls, *args, **kwargs)

Heredado de `GlueTransform` apply.

name(cls)

Heredado de `GlueTransform` name.

describeArgs(cls)

Heredado de `GlueTransform` describeArgs.

describeReturn(cls)

Heredado de `GlueTransform` describeReturn.

describeTransform(cls)

Heredado de `GlueTransform` describeTransform.

describeErrors(cls)

Heredado de `GlueTransform` describeErrors.

describe(cls)

Heredado de `GlueTransform` describe.

FormatPhoneNumber clase

La `FormatPhoneNumber` transformación devuelve una columna en la que la cadena de un número de teléfono se convierte en un valor formateado.

Ejemplo

```
from pyspark.context import SparkContext
from pyspark.sql import SparkSession
from awsgluedi.transforms import *

sc = SparkContext()
spark = SparkSession(sc)

input_df = spark.createDataFrame(
    [
        ("408-341-5669",),
        ("4083415669",)
    ],
    ["phone"],
)

try:
    df_output = column_formatting.FormatPhoneNumber.apply(
```

```
        data_frame=input_df,
        spark_context=sc,
        source_column="phone",
        default_region="US"
    )
    df_output.show()
except:
    print("Unexpected Error happened ")
    raise
```

Salida

El resultado será:

```
```
+---------------+
| phone|
+---------------+
|(408) 341-5669|
|(408) 341-5669|
+---------------+
```
```

La `FormatPhoneNumber` transformación toma la `source_column` como `"phone"` y la `default_region` como `"US"`.

La transformación formatea correctamente ambos números de teléfono, independientemente de su formato inicial, al formato estándar estadounidense `(408) 341-5669`.

Métodos

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__ (spark_context, data_frame, source_column, PHONE_NUMBER_FORMAT=NONE, DEFAULT_REGION=NONE, DEFAULT_REGION_COLUMN=NONE)

La transformación `FormatPhoneNumber` devuelve una columna en la que la cadena de un número de teléfono se convierte en un valor formateado.

- `source_column`: el nombre de una columna existente.
- `phone_number_format`— El formato al que se va a convertir el número de teléfono. Si no se especifica ningún formato, el formato predeterminado es `E.164` un formato de número de teléfono estándar reconocido internacionalmente. Entre los valores válidos se incluyen los siguientes:
  - E164 (omita el punto después de E)
- `default_region`— Un código de región válido compuesto por dos o tres letras mayúsculas que especifica la región del número de teléfono cuando no hay ningún código de país en el propio número. Como máximo, se `defaultRegionColumn` puede proporcionar uno de `defaultRegion` los siguientes.
- `default_region_column`— El nombre de una columna del tipo de datos avanzado`Country`. El código de región de la columna especificada se utiliza para determinar el código de país del número de teléfono cuando no hay ningún código de país en el propio número. Como máximo, se `defaultRegionColumn` puede proporcionar uno de `defaultRegion` los siguientes.

apply(cls, *args, **kwargs)

Heredado de `GlueTransform` apply.

name(cls)

Heredado de `GlueTransform` name.

describeArgs(cls)

Heredado de `GlueTransform` describeArgs.

describeReturn(cls)

Heredado de `GlueTransform` describeReturn.

describeTransform(cls)

Heredado de `GlueTransform` describeTransform.

describeErrors(cls)

Heredado de `GlueTransform` describeErrors.

describe(cls)

Heredado de `GlueTransform` describe.

FormatCase clase

La `FormatCase` transformación cambia cada cadena de una columna al tipo de mayúsculas y minúsculas especificado.

Ejemplo

```
from pyspark.context import SparkContext
from pyspark.sql import SparkSession
from awsgluedi.transforms import *

sc = SparkContext()
spark = SparkSession(sc)

datasource1 = spark.read.json("s3://${BUCKET}/json/zips/raw/data")

try:
    df_output = data_cleaning.FormatCase.apply(
        data_frame=datasource1,
        spark_context=sc,
        source_column="city",
        case_type="LOWER"
    )
except:
    print("Unexpected Error happened ")
    raise
```

Salida

La `FormatCase` transformación convertirá los valores de la columna `city` a minúsculas según el parámetro `CASE_TYPE="lower"`. La opción `df_output` resultante DataFrame contendrá todas las columnas de la `datasource1` original, pero con los valores de la columna `city` en minúsculas. DataFrame

Métodos

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__ (spark_context, data_frame, source_column, case_type)

La `FormatCase` transformación cambia cada cadena de una columna al tipo de mayúsculas y minúsculas especificado.

- `source_column`: el nombre de una columna existente.
- `case_type`— Los tipos de casos admitidos son `CAPITALLOWER`,`UPPER`,,`SENTENCE`.

apply(cls, *args, **kwargs)

Heredado de `GlueTransform` apply.

name(cls)

Heredado de `GlueTransform` name.

describeArgs(cls)

Heredado de `GlueTransform` describeArgs.

describeReturn(cls)

Heredado de `GlueTransform` describeReturn.

describeTransform(cls)

Heredado de `GlueTransform` describeTransform.

describeErrors(cls)

Heredado de `GlueTransform` describeErrors.

describe(cls)

Heredado de `GlueTransform` describe.

FillWithMode clase

La `FillWithMode` transformación formatea una columna de acuerdo con el formato de número de teléfono que especifique. También puede especificar una lógica de desempate, en la que algunos valores son idénticos. Por ejemplo, considere los siguientes valores: `1 2 2 3 3 4`

Un ModeType de `MINIMUM` hace `FillWithMode` que devuelva 2 como valor de modo. Si ModeType es`MAXIMUM`, el modo es 3. Para`AVERAGE`, el modo es 2.5.

Ejemplo

```
from awsglue.context import *
from pyspark.sql import SparkSession
from awsgluedi.transforms import *

sc = SparkContext()
spark = SparkSession(sc)

input_df = spark.createDataFrame(
    [
        (105.111, 13.12),
        (1055.123, 13.12),
        (None, 13.12),
        (13.12, 13.12),
        (None, 13.12),
    ],
    ["source_column_1", "source_column_2"],
)

try:
    df_output = data_quality.FillWithMode.apply(
        data_frame=input_df,
        spark_context=sc,
        source_column="source_column_1",
        mode_type="MAXIMUM"
    )
```

```
    df_output.show()
except:
    print("Unexpected Error happened ")
    raise
```

Salida

La salida del código dado será:

```
```
+---------------+---------------+
|source_column_1|source_column_2|
+---------------+---------------+
105.111	13.12
1055.123	13.12
1055.123	13.12
13.12	13.12
1055.123	13.12
+---------------+---------------+
```
```

La `FillWithMode` transformación del módulo `awsglue.data_quality` se aplica al `input_df`. DataFrame Sustituye los valores «nulos» de la columna por el valor máximo (`mode_TYPE="maximum"`) de los valores no nulos de esa columna. `source_column_1`

En este caso, el valor máximo de la columna es 1055,123. `source_column_1` Por lo tanto, los valores «nulos» de la salida se sustituyen por 1055.123 en la salida «`source_column_1df_output`». DataFrame

Métodos

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__ (spark_context, data_frame, source_column, mode_type)

La `FillWithMode` transformación formatea las cadenas de mayúsculas y minúsculas de una columna.

- `source_column`: el nombre de una columna existente.
- `mode_type`— Cómo resolver los valores de empate en los datos. Este valor debe ser uno de los `MINIMUM` siguientes:`NONE`,`AVERAGE`, o`MAXIMUM`.

apply(cls, *args, **kwargs)

Heredado de `GlueTransform` apply.

name(cls)

Heredado de `GlueTransform` name.

describeArgs(cls)

Heredado de `GlueTransform` describeArgs.

describeReturn(cls)

Heredado de `GlueTransform` describeReturn.

describeTransform(cls)

Heredado de `GlueTransform` describeTransform.

describeErrors(cls)

Heredado de `GlueTransform` describeErrors.

describe(cls)

Heredado de `GlueTransform` describe.

FlagDuplicateRows clase

La `FlagDuplicateRows` transformación devuelve una nueva columna con un valor específico en cada fila que indica si esa fila coincide exactamente con una fila anterior del conjunto de datos. Cuando se encuentran coincidencias, se marcan como duplicadas. La aparición inicial no está marcada porque no coincide con una fila anterior.

Ejemplo

```
from pyspark.context import SparkContext
from pyspark.sql import SparkSession
from awsgluedi.transforms import *

sc = SparkContext()
spark = SparkSession(sc)

input_df = spark.createDataFrame(
    [
        (105.111, 13.12),
        (13.12, 13.12),
        (None, 13.12),
        (13.12, 13.12),
        (None, 13.12),
    ],
    ["source_column_1", "source_column_2"],
)

try:
    df_output = data_quality.FlagDuplicateRows.apply(
        data_frame=input_df,
        spark_context=sc,
        target_column="flag_row",
        true_string="True",
        false_string="False",
        target_index=1
    )
except:
    print("Unexpected Error happened ")
    raise
```

Salida

El resultado será un PySpark DataFrame con una columna adicional `flag_row` que indica si una fila está duplicada o no, en función de la `source_column_1` columna. El `df_output` resultante DataFrame contendrá las siguientes filas:

```
```
+---------------+---------------+--------+
|source_column_1|source_column_2|flag_row|
```
```

```
+---------------+---------------+--------+
| 105.111| 13.12| False|
| 13.12| 13.12| True|
| null| 13.12| True|
| 13.12| 13.12| True|
| null| 13.12| True|
+---------------+---------------+--------+
```

La `flag_row` columna indica si una fila está duplicada o no. La `true_string` se establece en «True» y la `false_string` se establece en «False». El `target_index` está establecido en 1, lo que significa que la `flag_row` columna se insertará en la segunda posición (índice 1) de la salida. DataFrame

Métodos

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__ (spark_context, data_frame, target_column, true_string=DEFAULT_TRUE_STRING, FALSE_STRING=DEFAULT_FALSE_STRING, TARGET_INDEX=NONE)

La transformación devuelve una nueva columna con un valor específico en cada fila que indica si esa fila coincide exactamente con una fila anterior del conjunto de datos. `FlagDuplicateRows` Cuando se encuentran coincidencias, se marcan como duplicadas. La aparición inicial no está marcada porque no coincide con una fila anterior.

- `true_string`— Valor que se insertará si la fila coincide con una fila anterior.
- `false_string`— Valor que se insertará si la fila es única.
- `target_column`— Nombre de la nueva columna que se inserta en el conjunto de datos.

apply(cls, *args, **kwargs)

Heredado de `GlueTransform` apply.

name(cls)

Heredado de `GlueTransform` name.

describeArgs(cls)

Heredado de `GlueTransform` describeArgs.

describeReturn(cls)

Heredado de `GlueTransform` describeReturn.

describeTransform(cls)

Heredado de `GlueTransform` describeTransform.

describeErrors(cls)

Heredado de `GlueTransform` describeErrors.

describe(cls)

Heredado de `GlueTransform` describe.

RemoveDuplicates clase

La `RemoveDuplicates` transformación elimina una fila completa si se encuentra un valor duplicado en una columna de origen seleccionada.

Ejemplo

```
from pyspark.context import SparkContext
from pyspark.sql import SparkSession
from awsgluedi.transforms import *

sc = SparkContext()
spark = SparkSession(sc)

input_df = spark.createDataFrame(
    [
```

```
        (105.111, 13.12),
        (13.12, 13.12),
        (None, 13.12),
        (13.12, 13.12),
        (None, 13.12),
    ],
    ["source_column_1", "source_column_2"],
)

try:
    df_output = data_quality.RemoveDuplicates.apply(
        data_frame=input_df,
        spark_context=sc,
        source_column="source_column_1"
    )
except:
    print("Unexpected Error happened ")
    raise
```

Salida

El resultado será un y se eliminarán PySpark DataFrame los duplicados en función de la `source_column_1` columna. El `df_output` DataFrame resultante contendrá las siguientes filas:

```
+---------------+---------------+
|source_column_1|source_column_2|
+---------------+---------------+
| 105.111| 13.12|
| 13.12| 13.12|
| null| 13.12|
+---------------+---------------+
```

Ten en cuenta que las filas con `source_column_1` valores de 13,12 y nulos aparecen solo una vez en la salida, ya que los duplicados se han eliminado en función de la columna DataFrame. `source_column_1`

Métodos

- __call__

- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__ (spark_context, data_frame, source_column)

La `RemoveDuplicates` transformación elimina una fila completa si se encuentra un valor duplicado en una columna de origen seleccionada.

- `source_column`: el nombre de una columna existente.

apply(cls, *args, **kwargs)

Heredado de `GlueTransform` apply.

name(cls)

Heredado de `GlueTransform` name.

describeArgs(cls)

Heredado de `GlueTransform` describeArgs.

describeReturn(cls)

Heredado de `GlueTransform` describeReturn.

describeTransform(cls)

Heredado de `GlueTransform` describeTransform.

describeErrors(cls)

Heredado de `GlueTransform` describeErrors.

describe(cls)

Heredado de `GlueTransform` describe.

MonthName clase

La `MonthName` transformación crea una nueva columna que contiene el nombre del mes, a partir de una cadena que representa una fecha.

Ejemplo

```
from pyspark.context import SparkContext
from pyspark.sql import SparkSession
from awsgluedi.transforms import *

sc = SparkContext()
spark = SparkSession(sc)

spark.conf.set("spark.sql.legacy.timeParserPolicy", "LEGACY")

input_df = spark.createDataFrame(
    [
        ("20-2018-12",),
        ("2018-20-12",),
        ("20182012",),
        ("12202018",),
        ("20122018",),
        ("20-12-2018",),
        ("12/20/2018",),
        ("02/02/02",),
        ("02 02 2009",),
        ("02/02/2009",),
        ("August/02/2009",),
        ("02/june/2009",),
        ("02/2020/june",),
        ("2013-02-21 06:35:45.658505",),
        ("August 02 2009",),
        ("2013/02/21",),
        (None,),
    ],
    ["column_1"],
)
```

```
try:
    df_output = datetime_functions.MonthName.apply(
        data_frame=input_df,
        spark_context=sc,
        source_column="column_1",
        target_column="target_column"
    )
    df_output.show()
except:
    print("Unexpected Error happened ")
    raise
```

Salida

El resultado será:

```
```
+------------+------------+
| column_1|target_column|
+------------+------------+
20-2018-12	December
2018-20-12	null
20182012	null
12202018	null
20122018	null
20-12-2018	December
12/20/2018	December
02/02/02	February
02 02 2009	February
02/02/2009	February
August/02/2009	August
02/june/2009	null
02/2020/june	null
2013-02-21 06:35:45.658505	February
August 02 2009	August
2013/02/21	February
null	null
+------------+------------+
```
```

La MonthName transformación toma la `source_column` como `"column_1"` y la `target_column` como `"target_column"`. Intenta extraer el nombre del mes de las cadenas de fecha y hora de la

columna `"column_1"` y lo coloca en la columna `"target_column"`. Si la cadena de fecha y hora tiene un formato desconocido o no se puede analizar, el valor de `"target_column"` se establece en `null`.

La transformación extrae correctamente el nombre del mes de varios formatos de fecha y hora, como «20-12-2018», «20/12/2018", «02/02/2009", «2013-02-21 06:35:45 .658 505" y «02 de agosto de 2009".

Métodos

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__ (spark_context, data_frame, target_column, SOURCE_COLUMN=NONE, value=NONE)

La `MonthName` transformación crea una nueva columna que contiene el nombre del mes, a partir de una cadena que representa una fecha.

- `source_column`: el nombre de una columna existente.
- `value`— Una cadena de caracteres para evaluar..
- `target_column`— Un nombre para la columna recién creada.

apply(cls, *args, **kwargs)

Heredado de `GlueTransform` apply.

name(cls)

Heredado de `GlueTransform` name.

describeArgs(cls)

Heredado de `GlueTransform` describeArgs.

describeReturn(cls)

Heredado de `GlueTransform` describeReturn.

describeTransform(cls)

Heredado de `GlueTransform` describeTransform.

describeErrors(cls)

Heredado de `GlueTransform` describeErrors.

describe(cls)

Heredado de `GlueTransform` describe.

IsEven clase

La `IsEven` transformación devuelve un valor booleano en una nueva columna que indica si la columna o el valor de origen son pares. Si la columna o el valor de origen es decimal, el resultado es falso.

Ejemplo

```
from pyspark.context import SparkContext
from pyspark.sql import SparkSession
from awsgluedi.transforms import *

sc = SparkContext()
spark = SparkSession(sc)

input_df = spark.createDataFrame(
    [(5,), (0,), (-1,), (2,), (None,)],
    ["source_column"],
)

try:
    df_output = math_functions.IsEven.apply(
        data_frame=input_df,
        spark_context=sc,
        source_column="source_column",
        target_column="target_column",
        value=None,
```

```
        true_string="Even",
        false_string="Not even",
    )
    df_output.show()
except:
    print("Unexpected Error happened ")
    raise
```

Salida

El resultado será:

```
```
+-----------+-----------+
|source_column|target_column|
+-----------+-----------+
5	Not even
0	Even
-1	Not even
2	Even
null	null
+-----------+-----------+
```
```

La `IsEven` transformación toma la `source_column` como «source_column» y la `target_column` como «target_column». Comprueba si el valor de la `"columna_fuente"` es par o no. Si el valor es par, establece el valor de `"target_column"` en `true_string` «Even». Si el valor es impar, establece el valor de `"target_column"` en `false_string` «No es par». Si el valor de `"source_column"` es `nulo`, el valor de `"target_column"` se establece en `null`.

La transformación identifica correctamente los números pares (0 y 2) y establece el valor de `"target_column"` en «Even». Para los números impares (5 y -1), establece el valor de `"target_column"` en «No par». Para el valor `nulo` de `"source_column"`, el valor de `"target_column"` se establece en `null`.

Métodos

- __call__
- apply
- name

- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__ (spark_context, data_frame, target_column, SOURCE_COLUMN=NINGUNA, true_string=DEFAULT_TRUE_STRING, FALSE_STRING=DEFAULT_FALSE_STRING, value=NONE)

La transformación `IsEven` devuelve un valor booleano en una nueva columna que indica si la columna o el valor de origen son pares. Si la columna o el valor de origen es decimal, el resultado es falso.

- `source_column`: el nombre de una columna existente.
- `target_column`— El nombre de la nueva columna que se va a crear.
- `true_string`— Una cadena que indica si el valor es par.
- `false_string`— Una cadena que indica si el valor no es par.

apply(cls, *args, **kwargs)

Heredado de `GlueTransform` apply.

name(cls)

Heredado de `GlueTransform` name.

describeArgs(cls)

Heredado de `GlueTransform` describeArgs.

describeReturn(cls)

Heredado de `GlueTransform` describeReturn.

describeTransform(cls)

Heredado de `GlueTransform` describeTransform.

describeErrors(cls)

Heredado de `GlueTransform` describeErrors.

describe(cls)

Heredado de `GlueTransform` describe.

CryptographicHash clase

La `CryptographicHash` transformación aplica un algoritmo a los valores de hash de la columna.

Ejemplo

```
from pyspark.context import SparkContext
from pyspark.sql import SparkSession
from awsgluedi.transforms import *

secret = "${SECRET}"
sc = SparkContext()
spark = SparkSession(sc)

input_df = spark.createDataFrame(
    [
        (1, "1234560000"),
        (2, "1234560001"),
        (3, "1234560002"),
        (4, "1234560003"),
        (5, "1234560004"),
        (6, "1234560005"),
        (7, "1234560006"),
        (8, "1234560007"),
        (9, "1234560008"),
        (10, "1234560009"),
    ],
    ["id", "phone"],
)

try:
    df_output = pii.CryptographicHash.apply(
        data_frame=input_df,
        spark_context=sc,
        source_columns=["id", "phone"],
        secret_id=secret,
```

```
        algorithm="HMAC_SHA256",
        output_format="BASE64",
    )
    df_output.show()
except:
    print("Unexpected Error happened ")
    raise
```

Salida

El resultado será:

```
```
+---+------------+------------------+------------------+
| id| phone | id_hashed | phone_hashed |
+---+------------+------------------+------------------+
1	1234560000	QUI1zXTJiXmfIb...	juDBAmiRnnO3g...
2	1234560001	ZAUWiZ3dVTzCo...	vC8lgUqBVDMNQ...
3	1234560002	ZP4VvZWkqYifu...	Kl3QAkgswYpzB...
4	1234560003	3u8vO3wQ8EQfj...	CPBzK1P8PZZkV...
5	1234560004	eWkQJk4zAOIzx...	aLf7+mHcXqbLs...
6	1234560005	xtI9fZCJZCvsa...	dy2DFgdYWmr0p...
7	1234560006	iW9hew7jnHuOf...	wwfGMCOEv6oOv...
8	1234560007	H9V1pqvgkFhfS...	g9WKhagIXy9ht...
9	1234560008	xDhEuHaxAUbU5...	b3uQLKPY+Q5vU...
10	1234560009	GRN6nFXkxk349...	VJdsKt8VbxBbt...
+---+------------+------------------+------------------+
```
```

La transformación calcula los hashes criptográficos de los valores de las columnas `id` y `phone` utilizando el algoritmo y la clave secreta especificados, y codifica los hashes en formato Base64. La opción `df_output` resultante DataFrame contiene todas las columnas de la `input_df` original, además de las columnas adicionales `id_hashed` y `phone_hashed` con los valores hash calculados. DataFrame

Métodos

- __call__
- apply
- name

- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__ (spark_context, data_frame, source_columns, secret_id, algorithm=None, secret_version=None, create_secret_if_missing=false, output_format=Ninguno, entity_type_filter=Ninguno)

La transformación `CryptographicHash` aplica un algoritmo a los valores de hash de la columna.

- `source_columns`— Un conjunto de columnas existentes.
- `secret_id`— El ARN de la clave secreta de Secrets Manager. La clave utilizada en el algoritmo de prefijo del código de autenticación de mensajes (HMAC) basado en hash para codificar las columnas de origen.
- `secret_version`: opcional. De forma predeterminada, es la última versión secreta.
- `entity_type_filter`— Matriz opcional de tipos de entidades. Se puede usar para cifrar solo la PII detectada en la columna de texto libre.
- `create_secret_if_missing`— Booleano opcional. Si es verdadero, intentará crear el secreto en nombre de la persona que llama.
- `algorithm`— El algoritmo utilizado para codificar sus datos. Valores de enumeración válidos: MD5, SHA1, SHA256, SHA512, HMAC_MD5, HMAC_SHA1, HMAC_SHA256, HMAC_SHA512.

apply(cls, *args, **kwargs)

Heredado de `GlueTransform` apply.

name(cls)

Heredado de `GlueTransform` name.

describeArgs(cls)

Heredado de `GlueTransform` describeArgs.

describeReturn(cls)

Heredado de `GlueTransform` describeReturn.

describeTransform(cls)

Heredado de `GlueTransform` describeTransform.

describeErrors(cls)

Heredado de `GlueTransform` describeErrors.

describe(cls)

Heredado de `GlueTransform` describe.

Clase de desencriptación

La `Decrypt` transformación descifra el interior de AWS Glue. Sus datos también se pueden descifrar fuera de AWS Glue con el SDK de AWS cifrado. Si la clave de KMS ARN proporcionada no coincide con la utilizada para cifrar la columna, se produce un error en la operación de descifrado.

Ejemplo

```
from pyspark.context import SparkContext
from pyspark.sql import SparkSession
from awsgluedi.transforms import *

kms = "${KMS}"
sc = SparkContext()
spark = SparkSession(sc)

input_df = spark.createDataFrame(
    [
        (1, "1234560000"),
        (2, "1234560001"),
        (3, "1234560002"),
        (4, "1234560003"),
        (5, "1234560004"),
        (6, "1234560005"),
        (7, "1234560006"),
        (8, "1234560007"),
        (9, "1234560008"),
        (10, "1234560009"),
    ],
```

```
    ["id", "phone"],
)

try:
    df_encrypt = pii.Encrypt.apply(
        data_frame=input_df,
        spark_context=sc,
        source_columns=["phone"],
        kms_key_arn=kms
    )
    df_decrypt = pii.Decrypt.apply(
        data_frame=df_encrypt,
        spark_context=sc,
        source_columns=["phone"],
        kms_key_arn=kms
    )
    df_decrypt.show()
except:
    print("Unexpected Error happened ")
    raise
```

Salida

El resultado será una PySpark DataFrame con la columna `id` original y la columna `teléfono` descifrada:

```
```
+---+------------+
| id| phone|
+---+------------+
1	1234560000
2	1234560001
3	1234560002
4	1234560003
5	1234560004
6	1234560005
7	1234560006
8	1234560007
9	1234560008
10	1234560009
+---+------------+
```
```

La `Encrypt` transformación toma las `source_columns` como `["phone"]` y `kms_key_arn` como el valor de la variable de entorno `$ {KMS} `. La transformación cifra los valores de la columna `phone` mediante la clave KMS especificada. A continuación, el DataFrame `df_encrypt` cifrado se pasa a la transformación desde el módulo `awsglue.pii`. `Decrypt` Toma las `source_columns` como `["phone"]` y `kms_key_arn` como el valor de la variable de entorno `$ {KMS} `. La transformación descifra los valores cifrados de la columna `phone` utilizando la misma clave KMS. El `df_decrypt` resultante DataFrame contiene la columna `id` original y la columna `teléfono` descifrada.

Métodos

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__ (spark_context, data_frame, source_columns, kms_key_arn)

La `Decrypt` transformación descifra el interior de AWS Glue. Sus datos también se pueden descifrar fuera de AWS Glue con el SDK de AWS cifrado. Si la clave de KMS ARN proporcionada no coincide con la utilizada para cifrar la columna, se produce un error en la operación de descifrado.

- `source_columns`— Matriz de columnas existentes.
- `kms_key_arn`— La clave ARN de la clave del Servicio de administración de AWS claves que se utilizará para descifrar las columnas de origen.

apply(cls, *args, **kwargs)

Heredado de `GlueTransform` apply.

name(cls)

Heredado de `GlueTransform` name.

describeArgs(cls)

Heredado de `GlueTransform` describeArgs.

describeReturn(cls)

Heredado de `GlueTransform` describeReturn.

describeTransform(cls)

Heredado de `GlueTransform` describeTransform.

describeErrors(cls)

Heredado de `GlueTransform` describeErrors.

describe(cls)

Heredado de `GlueTransform` describe.

Clase de cifrado

La `Encrypt` transformación cifra las columnas de origen mediante la clave del Servicio de administración de AWS claves. La `Encrypt` transformación puede cifrar hasta 128 MiB por celda. Intentará conservar el formato al descifrarlo. Para conservar el tipo de datos, los metadatos del tipo de datos deben serializarse a menos de 1 KB. De lo contrario, debe establecer el `preserve_data_type` parámetro en false. Los metadatos del tipo de datos se almacenarán en texto plano en el contexto de cifrado.

Ejemplo

```
from pyspark.context import SparkContext
from pyspark.sql import SparkSession
from awsgluedi.transforms import *

kms = "${KMS}"
sc = SparkContext()
spark = SparkSession(sc)

input_df = spark.createDataFrame(
    [
        (1, "1234560000"),
        (2, "1234560001"),
        (3, "1234560002"),
```

```
        (4, "1234560003"),
        (5, "1234560004"),
        (6, "1234560005"),
        (7, "1234560006"),
        (8, "1234560007"),
        (9, "1234560008"),
        (10, "1234560009"),
    ],
    ["id", "phone"],
)

try:
    df_encrypt = pii.Encrypt.apply(
        data_frame=input_df,
        spark_context=sc,
        source_columns=["phone"],
        kms_key_arn=kms
    )
except:
    print("Unexpected Error happened ")
    raise
```

Salida

El resultado será una PySpark DataFrame con la columna `id` original y una columna adicional que contendrá los valores cifrados de la columna `teléfono`.

```
```
+---+------------+------------------------+
| id| phone | phone_encrypted |
+---+------------+------------------------+
1	1234560000	EncryptedData1234...abc
2	1234560001	EncryptedData5678...def
3	1234560002	EncryptedData9012...ghi
4	1234560003	EncryptedData3456...jkl
5	1234560004	EncryptedData7890...mno
6	1234560005	EncryptedData1234...pqr
7	1234560006	EncryptedData5678...stu
8	1234560007	EncryptedData9012...vwx
9	1234560008	EncryptedData3456...yz0
10	1234560009	EncryptedData7890...123
+---+------------+------------------------+
```
```

```
```

La `Encrypt` transformación toma `source_columns` como `["phone"]` y `kms_key_arn` como el valor de la variable de entorno `$ {KMS} `. La transformación cifra los valores de la columna `phone` mediante la clave KMS especificada. El `df_encrypt` resultante DataFrame contiene la columna `id` original, la columna `teléfono` original y una columna adicional denominada `phone_encrypted` que contiene los valores cifrados de la columna `teléfono`.

Métodos

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__ (spark_context, data_frame, source_columns, kms_key_arn, entity_type_filter=None, preserve_data_type=None)

La transformación `Encrypt` cifra las columnas de origen mediante la AWS clave del Servicio de administración de claves.

- `source_columns`— Un conjunto de columnas existentes.
- `kms_key_arn`— La clave ARN de la clave del Servicio de administración de AWS claves que se utilizará para cifrar las columnas de origen.
- `entity_type_filter`— Matriz opcional de tipos de entidades. Se puede usar para cifrar solo la PII detectada en la columna de texto libre.
- `preserve_data_type`— Booleano opcional. El valor predeterminado es true (verdadero). Si es falso, el tipo de datos no se almacenará.

apply(cls, *args, **kwargs)

Heredado de `GlueTransform` apply.

name(cls)

Heredado de `GlueTransform` name.

describeArgs(cls)

Heredado de `GlueTransform` describeArgs.

describeReturn(cls)

Heredado de `GlueTransform` describeReturn.

describeTransform(cls)

Heredado de `GlueTransform` describeTransform.

describeErrors(cls)

Heredado de `GlueTransform` describeErrors.

describe(cls)

Heredado de `GlueTransform` describe.

IntToIp clase

La `IntToIp` transformación convierte el valor entero de la columna de origen u otro valor en el valor IPv4 correspondiente de la columna de destino y devuelve el resultado en una nueva columna.

Ejemplo

```
from pyspark.context import SparkContext
from pyspark.sql import SparkSession
from awsgluedi.transforms import *

sc = SparkContext()
spark = SparkSession(sc)

input_df = spark.createDataFrame(
    [
        (3221225473,),
        (0,),
        (1,),
        (100,),
        (168430090,),
```

```
        (4294967295,),
        (4294967294,),
        (4294967296,),
        (-1,),
        (None,),
    ],
    ["source_column_int"],
)

try:
    df_output = web_functions.IntToIp.apply(
        data_frame=input_df,
        spark_context=sc,
        source_column="source_column_int",
        target_column="target_column",
        value=None
    )
    df_output.show()
except:
    print("Unexpected Error happened ")
    raise
```

Salida

El resultado será:

```
```
+---------------+---------------+
|source_column_int|target_column|
+---------------+---------------+
3221225473	192.0.0.1
0	0.0.0.0
1	0.0.0.1
100	0.0.0.100
168430090	10.0.0.10
4294967295	255.255.255.255
4294967294	255.255.255.254
4294967296	null
-1	null
null	null
+---------------+---------------+
```
```

La `IntToIp.apply` transformación toma la `source_column` como `"'source_column_int"'` y la `target_column` como `"'target_column"'` y convierte los valores enteros de la columna `source_column_int` en su representación de dirección IPv4 correspondiente y almacena el resultado en la columna `target_column`.

Para valores enteros válidos dentro del rango de direcciones IPv4 (0 a 4294967295), la transformación los convierte correctamente en su representación de direcciones IPv4 (por ejemplo, 192.0.0.1, 0.0.0.0, 10.0.0.10, 255.255.255.255).

Para valores enteros fuera del rango válido (por ejemplo, 4294967296, -1), el valor `target_column` se establece en `null`. Para los valores «nulos» de la columna «source_column_int», el valor de «target_column» también se establece en «nulo».

Métodos

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__ (spark_context, data_frame, target_column, source_column=None, value=NONE)

La `IntToIp` transformación convierte el valor entero de la columna de origen u otro valor en el valor IPv4 correspondiente de la columna de destino y devuelve el resultado en una nueva columna.

- `sourceColumn`: el nombre de una columna existente.
- `value`— Una cadena de caracteres para evaluar.
- `targetColumn`— El nombre de la nueva columna que se va a crear.

apply(cls, *args, **kwargs)

Heredado de `GlueTransform` apply.

name(cls)

Heredado de `GlueTransform` name.

describeArgs(cls)

Heredado de `GlueTransform` describeArgs.

describeReturn(cls)

Heredado de `GlueTransform` describeReturn.

describeTransform(cls)

Heredado de `GlueTransform` describeTransform.

describeErrors(cls)

Heredado de `GlueTransform` describeErrors.

describe(cls)

Heredado de `GlueTransform` describe.

IpToInt clase

La `IpToInt` transformación convierte el valor del Protocolo de Internet versión 4 (IPv4) de la columna de origen u otro valor en el valor entero correspondiente de la columna de destino y devuelve el resultado en una nueva columna.

Ejemplo

Para AWS Glue 4.0 y versiones posteriores, cree o actualice argumentos de trabajo con `key: --enable-glue-di-transforms, value: true`

```
from pyspark.context import SparkContext
from awsgluedi.transforms import *

sc = SparkContext()

input_df = spark.createDataFrame(
    [
```

```
        ("192.0.0.1",),
        ("10.10.10.10",),
        ("1.2.3.4",),
        ("1.2.3.6",),
        ("http://12.13.14.15",),
        ("https://16.17.18.19",),
        ("1.2.3.4",),
        (None,),
        ("abc",),
        ("abc.abc.abc.abc",),
        ("321.123.123.123",),
        ("244.4.4.4",),
        ("255.255.255.255",),
    ],
    ["source_column_ip"],
)

    df_output = web_functions.IpToInt.apply(
        data_frame=input_df,
        spark_context=sc,
        source_column="source_column_ip",
        target_column="target_column",
        value=None
    )
    df_output.show()
```

Salida

El resultado será:

```
```
+----------------+---------------+
|source_column_ip| target_column|
+----------------+---------------+
192.0.0.1	3221225473
10.10.10.10	168427722
1.2.3.4	16909060
1.2.3.6	16909062
http://12.13.14.15	null
https://16.17.18.19	null
1.2.3.4	16909060
null	null
abc	null
```
```

```
|abc.abc.abc.abc| null|
| 321.123.123.123| null|
| 244.4.4.4| 4102444804|
| 255.255.255.255| 4294967295|
+----------------+---------------+
```

La `IpToInt` transformación toma la `source_column` como `"source_column_ip"` y la `target_column` como `"target_column"` y convierte las cadenas de direcciones IPv4 válidas de la columna `source_column_ip` en su correspondiente representación entera de 32 bits y almacena el resultado en la columna `target_column`.

Para cadenas de direcciones IPv4 válidas (por ejemplo, «192.0.0.1", «10.10.10", «1.2.3.4»), la transformación las convierte correctamente en su representación de números enteros (por ejemplo, 3221225473, 168427722, 16909060). Para las cadenas que no son direcciones IPv4 válidas (por ejemplo, direcciones URL, cadenas que no son IP como «abc» o formatos IP no válidos como «abc.abc.abc.abc»), el valor de `target_column` se establece en `null`. Para los valores `nulos» de la columna `source_column_ip`, el valor `target_column` también se establece en `null`.

Métodos

- __call__
- apply
- name
- describeArgs
- describeReturn
- describeTransform
- describeErrors
- describe

__call__ (spark_context, data_frame, target_column, source_column=None, value=NONE)

La `IpToInt` transformación convierte el valor del Protocolo de Internet versión 4 (IPv4) de la columna de origen u otro valor en el valor entero correspondiente de la columna de destino y devuelve el resultado en una nueva columna.

- `sourceColumn`: el nombre de una columna existente.

- `value`— Una cadena de caracteres para evaluar.
- `targetColumn`— El nombre de la nueva columna que se va a crear.

apply(cls, *args, **kwargs)

Heredado de `GlueTransform` apply.

name(cls)

Heredado de `GlueTransform` name.

describeArgs(cls)

Heredado de `GlueTransform` describeArgs.

describeReturn(cls)

Heredado de `GlueTransform` describeReturn.

describeTransform(cls)

Heredado de `GlueTransform` describeTransform.

describeErrors(cls)

Heredado de `GlueTransform` describeErrors.

describe(cls)

Heredado de `GlueTransform` describe.

Transformaciones de integración de datos

Para AWS Glue 4.0 y versiones posteriores, cree o actualice argumentos de trabajo con`key: --enable-glue-di-transforms, value: true`.

Ejemplo de script de trabajo:

```
from pyspark.context import SparkContext
```

```
from awsgluedi.transforms import *
sc = SparkContext()

input_df = spark.createDataFrame(
    [(5,), (0,), (-1,), (2,), (None,)],
    ["source_column"],
)

try:
    df_output = math_functions.IsEven.apply(
        data_frame=input_df,
        spark_context=sc,
        source_column="source_column",
        target_column="target_column",
        value=None,
        true_string="Even",
        false_string="Not even",
    )
    df_output.show()
except:
    print("Unexpected Error happened ")
    raise
```

Ejemplos de sesiones con cuadernos

```
%idle_timeout 2880
%glue_version 4.0
%worker_type G.1X
%number_of_workers 5
%region eu-west-1
```

```
%%configure
{
    "--enable-glue-di-transforms": "true"
}
```

```
from pyspark.context import SparkContext
from awsgluedi.transforms import *

sc = SparkContext()

input_df = spark.createDataFrame(
```

```
    [(5,), (0,), (-1,), (2,), (None,)],
    ["source_column"],
)

try:
    df_output = math_functions.IsEven.apply(
        data_frame=input_df,
        spark_context=sc,
        source_column="source_column",
        target_column="target_column",
        value=None,
        true_string="Even",
        false_string="Not even",
    )
    df_output.show()
except:
    print("Unexpected Error happened ")
    raise
```

Ejemplos de sesiones utilizando AWS CLI

```
aws glue create-session --default-arguments "--enable-glue-di-transforms=true"
```

Transformaciones DI:

- FlagDuplicatesInColumn clase
- FormatPhoneNumber clase
- FormatCase clase
- FillWithMode clase
- FlagDuplicateRows clase
- RemoveDuplicates clase
- MonthName clase
- IsEven clase
- CryptographicHash clase
- Clase de desencriptación
- Clase de cifrado
- IntToIp clase

- IpToInt clase

Maven: combina el plugin con tus aplicaciones de Spark

Puedes agrupar la dependencia de las transformaciones con tus aplicaciones y distribuciones de Spark (versión 3.3) añadiendo la dependencia del plugin en tu Maven `pom.xml` y desarrollando tus aplicaciones de Spark de forma local.

```
<repositories>
   ...
    <repository>
        <id>aws-glue-etl-artifacts</id>
        <url>https://aws-glue-etl-artifacts.s3.amazonaws.com/release/ </url>
    </repository>
</repositories>
...
<dependency>
    <groupId>com.amazonaws</groupId>
    <artifactId>AWSGlueTransforms</artifactId>
    <version>4.0.0</version>
</dependency>
```

También puedes descargar directamente los binarios de los artefactos de AWS Glue Maven e incluirlos en tu aplicación Spark de la siguiente manera.

```
#!/bin/bash
sudo wget -v https://aws-glue-etl-artifacts.s3.amazonaws.com/release/com/amazonaws/
AWSGlueTransforms/4.0.0/AWSGlueTransforms-4.0.0.jar -P /usr/lib/spark/jars/
```

Programación de scripts de ETL de AWS Glue en Scala

Puede encontrar ejemplos de código de Scala y utilidades para AWS Glue en el repositorio de ejemplos de AWS Glue en el sitio web de GitHub.

AWS Glue soporta una extensión del dialecto PySpark Scala para scripting de trabajos de extracción, transformación y carga (ETL). En las secciones siguientes se describe cómo utilizar la biblioteca de AWS Glue Scala y la API de AWS Glue en los scripts de ETL y se proporciona documentación de referencia correspondiente a la biblioteca.

Contenido

- Uso de Scala para programar scripts de ETL de AWS Glue
  - Prueba de un programa ETL de Scala en un cuaderno de Jupyter en un punto de conexión de desarrollo
  - Prueba de un programa ETL de Scala en un REPL de Scala
- Ejemplo de script de Scala - ETL de streaming
- API en la biblioteca Scala de AWS Glue
  - com.amazonaws.services.glue
  - com.amazonaws.services.glue.ml
  - com.amazonaws.services.glue.dq
  - com.amazonaws.services.glue.types
  - com.amazonaws.services.glue.util
  - API ChoiceOption Scala de AWS Glue
    - Característica ChoiceOption
    - Objeto ChoiceOption
      - Def apply
    - case class ChoiceOptionWithResolver
    - case class MatchCatalogSchemaChoiceOption
  - Abstract DataSink class
    - def writeDynamicFrame
    - def pyWriteDynamicFrame
    - Def writeDataFrame
    - Def pyWriteDataFrame
    - def setCatalogInfo
    - def supportsFormat
    - def setFormat
    - def withFormat
    - def setAccumulableSize
    - def getOutputErrorRecordsAccumulable
    - def errorsAsDynamicFrame
    - Objeto DataSink
      - def recordMetrics

- Característica DataSource Scala de AWS Glue
- API DynamicFrame Scala de AWS Glue
  - Clase DynamicFrame Scala de AWS Glue
    - Val errorsCount
    - Def applyMapping
    - Def assertErrorThreshold
    - Def count
    - Def dropField
    - Def dropFields
    - Def dropNulls
    - def errorsAsDynamicFrame
    - Def filter
    - Def getName
    - Def getNumPartitions
    - Def getSchemaIfComputed
    - Def isSchemaComputed
    - Def javaToPython
    - Def join
    - Def map
    - Def mergeDynamicFrames
    - Def printSchema
    - Def recomputeSchema
    - Def relationalize
    - Def renameField
    - Def repartition
    - Def resolveChoice
    - Def schema
    - Def selectField
    - Def selectFields
    - Def show

        - Def simplifyDDBJson
        - Def spigot
        - Def splitFields
        - Def splitRows
        - Def stageErrorsCount
        - Def toDF
        - Def unbox
        - Def unnest
        - Def unnestDDBJson
        - Def withFrameSchema
        - Def withName
        - Def withTransformationContext
    - Objeto DynamicFrame
        - Def apply
        - def emptyDynamicFrame
        - Def fromPythonRDD
        - Def ignoreErrors
        - Def inlineErrors
        - Def newFrameWithErrors
- Clase DynamicRecord Scala de AWS Glue
    - Def addField
    - Def dropField
    - Def setError
    - Def isError
    - Def getError
    - Def clearError
    - Def write
    - Def readFields
    - Def clone

    - Def schema

- Def getRoot
- Def toJson
- Def getFieldNode
- Def getField
- Def hashCode
- Def equals
- Objeto DynamicRecord
  - Def apply
- Característica RecordTraverser

- AWS GlueAPI de Scala GlueContext
  - columnas def addIngestionTime
  - def createDataFrame FromOptions
  - forEachBatch
  - - definitivamente getCatalogSink
  - def getCatalogSource
  - def getJDBCSink
  - def getSink
  - Formato def getSinkWith
  - def getSource
  - Formato def getSourceWith
  - def getSparkSession
  - Definición de startTransaction (iniciar transacción)
  - Definición de commitTransaction (confirmar transacción)
  - Definición de cancelTransaction (cancelar transacción)
  - def this
  - def this
  - def this

- MappingSpec
  - Case class MappingSpec
  - Objeto MappingSpec

- Val orderingByTarget
- Def apply
- Def apply
- Def apply

- API ResolveSpec Scala de AWS Glue
  - Objeto ResolveSpec
    - Def
    - Def
  - Case class ResolveSpec
    - Métodos def de ResolveSpec
- API ArrayNode Scala de AWS Glue
  - Case class ArrayNode
    - Métodos def de ArrayNode
- API BinaryNode Scala de AWS Glue
  - Case class BinaryNode
    - Campos val de BinaryNode
    - Métodos def de BinaryNode
- API BooleanNode Scala de AWS Glue
  - Case class BooleanNode
    - Campos val de BooleanNode
    - Métodos def de BooleanNode
- API ByteNode Scala de AWS Glue
  - Case class ByteNode
    - Campos val de ByteNode
    - Métodos def de ByteNode
- API DateNode Scala de AWS Glue
  - Case class DateNode
    - Campos val de DateNode
    - Métodos def de DateNode

- API DecimalNode Scala de AWS Glue

    - Case class DecimalNode
        - Campos val de DecimalNode
        - Métodos def de DecimalNode
- API DoubleNode Scala de AWS Glue
    - Case class DoubleNode
        - Campos val de DoubleNode
        - Métodos def de DoubleNode
- API DynamicNode Scala de AWS Glue
    - Clase DynamicNode
        - Métodos def de DynamicNode
    - Objeto DynamicNode
        - Métodos def de DynamicNode
- Clase de EvaluateDataQuality
    - Def apply
    - Ejemplo
- API FloatNode Scala de AWS Glue
    - Case class FloatNode
        - Campos val de FloatNode
        - Métodos def de FloatNode
- Clase FillMissingValues
    - Def apply
- Clase FindMatches
    - Def apply
- Clase FindIncrementalMatches
    - Def apply
- API IntegerNode Scala de AWS Glue
    - Case class IntegerNode
        - Campos val de IntegerNode
        - Métodos def de IntegerNode
- API LongNode Scala de AWS Glue

    - Case class LongNode
        - Campos val de LongNode
        - Métodos def de LongNode
- API MapLikeNode Scala de AWS Glue
    - Case class MapLikeNode
        - Métodos def de MapLikeNode
- API MapNode Scala de AWS Glue
    - Case class MapNode
        - Métodos def de MapNode
- API NullNode Scala de AWS Glue
    - Clase NullNode
    - Case object NullNode
- API ObjectNode Scala de AWS Glue
    - Objeto ObjectNode
        - Métodos def de ObjectNode
    - Case class ObjectNode
        - Métodos def de ObjectNode
- API ScalarNode Scala de AWS Glue
    - Clase ScalarNode
        - Métodos def de ScalarNode
    - Objeto ScalarNode
        - Métodos def de ScalarNode
- API ShortNode Scala de AWS Glue
    - Case class ShortNode
        - Campos val de ShortNode
        - Métodos def de ShortNode
- API StringNode Scala de AWS Glue
    - Case class StringNode
        - Campos val de StringNode
        - Métodos def de StringNode

- API TimestampNode Scala de AWS Glue
  - Case class TimestampNode
    - Campos val de TimestampNode
    - Métodos def de TimestampNode
- API GlueArgParser Scala de AWS Glue
  - Objeto GlueArgParser
    - Métodos def de GlueArgParser
- API Job Scala de AWS Glue
  - Objeto de trabajo
    - Métodos def de trabajo

Uso de Scala para programar scripts de ETL de AWS Glue

Puede generar automáticamente un programa de extracción, transformación y carga (ETL) Scala usando la consola de AWS Glue y modificándolo según sea necesario antes de asignarlo a un flujo de trabajo. O bien puede escribir su propio programa desde cero. Para obtener más información, consulte Configurar las propiedades de los trabajos de Spark en AWS Glue. AWS Glue compila el programa de Scala en el servidor antes de ejecutar el flujo de trabajo asociado.

Para asegurarse de que el programa se compila sin errores y se ejecuta según lo previsto, es muy importante que lo cargue en un punto de conexión de desarrollo en REPL (Read-Eval-Print Loop) o un cuaderno de Jupyter y que lo pruebe antes de ejecutarlo en un flujo de trabajo. Debido a que el proceso de compilación se realiza en el servidor, no tendrá una visibilidad adecuada de los problemas que se produzcan en él.

Prueba de un programa ETL de Scala en un cuaderno de Jupyter en un punto de conexión de desarrollo

Para probar un programa Scala en un enlace de desarrollo de AWS Glue, configure el punto de enlace de desarrollo tal y como se describe en Añadir un punto de conexión de desarrollo..

A continuación, conéctelo a un cuaderno de Jupyter que se ejecute localmente en la máquina o de forma remota en un servidor de cuadernos de Amazon EC2. Para instalar una versión local de un cuaderno de Jupyter, siga las instrucciones de Tutorial: cuaderno de Jupyter en JupyterLab.

La única diferencia entre ejecutar código de Scala y ejecutar código de PySpark en su bloc de notas es que debe comenzar cada párrafo del bloc de notas con lo siguiente:

```
%spark
```

De este modo se impide que el servidor de bloc de notas cambie de forma predeterminada a la versión PySpark del intérprete de Spark.

Prueba de un programa ETL de Scala en un REPL de Scala

Puede probar un programa Scala en un punto de enlace de desarrollo con un REPL de Scala de AWS Glue. Siga las instrucciones de Tutorial: Uso de un bloc de notas de SageMaker, excepto al final del comando SSH-to-REPL, reemplace `-t gluepyspark` por `-t glue-spark-shell`. Esto invoca el REPL de Scala de AWS Glue.

Para cerrar el REPL cuando termine, escriba `sys.exit`.

Ejemplo de script de Scala - ETL de streaming

Example

El siguiente script de ejemplo se conecta a Amazon Kinesis Data Streams, utiliza un esquema del Data Catalog para analizar un flujo de datos, une el flujo a un conjunto de datos estático en Amazon S3 y envía los resultados combinados a Amazon S3 en formato parquet.

```
// This script connects to an Amazon Kinesis stream, uses a schema from the data
 catalog to parse the stream,
// joins the stream to a static dataset on Amazon S3, and outputs the joined results to
 Amazon S3 in parquet format.
import com.amazonaws.services.glue.GlueContext
import com.amazonaws.services.glue.util.GlueArgParser
import com.amazonaws.services.glue.util.Job
import java.util.Calendar
import org.apache.spark.SparkContext
import org.apache.spark.sql.Dataset
import org.apache.spark.sql.Row
import org.apache.spark.sql.SaveMode
import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.functions.from_json
import org.apache.spark.sql.streaming.Trigger
import scala.collection.JavaConverters._

object streamJoiner {
  def main(sysArgs: Array[String]) {
    val spark: SparkContext = new SparkContext()
```

```
    val glueContext: GlueContext = new GlueContext(spark)
    val sparkSession: SparkSession = glueContext.getSparkSession
    import sparkSession.implicits._
    // @params: [JOB_NAME]
    val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB_NAME").toArray)
    Job.init(args("JOB_NAME"), glueContext, args.asJava)

    val staticData = sparkSession.read          // read() returns type DataFrameReader
      .format("csv")
      .option("header", "true")
      .load("s3://awsexamplebucket-streaming-demo2/inputs/productsStatic.csv")  //
load() returns a DataFrame

    val datasource0 = sparkSession.readStream   // readstream() returns type
DataStreamReader
      .format("kinesis")
      .option("streamName", "stream-join-demo")
      .option("endpointUrl", "https://kinesis.us-east-1.amazonaws.com")
      .option("startingPosition", "TRIM_HORIZON")
      .load                                     // load() returns a DataFrame

    val selectfields1 = datasource0.select(from_json($"data".cast("string"),
glueContext.getCatalogSchemaAsSparkSchema("stream-demos", "stream-join-demo2")) as
"data").select("data.*")

    val datasink2 = selectfields1.writeStream.foreachBatch { (dataFrame: Dataset[Row],
batchId: Long) => {   //foreachBatch() returns type DataStreamWriter
      val joined = dataFrame.join(staticData, "product_id")
      val year: Int = Calendar.getInstance().get(Calendar.YEAR)
      val month :Int = Calendar.getInstance().get(Calendar.MONTH) + 1
      val day: Int = Calendar.getInstance().get(Calendar.DATE)
      val hour: Int = Calendar.getInstance().get(Calendar.HOUR_OF_DAY)

      if (dataFrame.count() > 0) {
        joined.write                            // joined.write returns type
DataFrameWriter
          .mode(SaveMode.Append)
          .format("parquet")
          .option("quote", " ")
          .save("s3://awsexamplebucket-streaming-demo2/output/" + "/year=" +
"%04d".format(year) + "/month=" + "%02d".format(month) + "/day=" + "%02d".format(day)
+ "/hour=" + "%02d".format(hour) + "/")
      }
    }
```

```
    }  // end foreachBatch()
      .trigger(Trigger.ProcessingTime("100 seconds"))
      .option("checkpointLocation", "s3://awsexamplebucket-streaming-demo2/
checkpoint/")
      .start().awaitTermination()              // start() returns type StreamingQuery
    Job.commit()
  }
}
```

API en la biblioteca Scala de AWS Glue

AWS Glue es compatible con una extensión del dialecto Scala de PySpark para los trabajos de extraer, transformar y cargar (ETL) scripts. En las siguientes secciones se describen las API de la biblioteca Scala de AWS Glue.

com.amazonaws.services.glue

El paquete com.amazonaws.services.glue de la biblioteca AWS Glue Scala contiene las API siguientes:

- ChoiceOption
- DataSink
- Característica DataSource
- DynamicFrame
- DynamicRecord
- GlueContext
- MappingSpec
- ResolveSpec

com.amazonaws.services.glue.ml

El paquete com.amazonaws.services.glue.ml de la biblioteca de AWS Glue Scala contiene las siguientes API:

- FillMissingValues
- FindIncrementalMatches
- FindMatches

com.amazonaws.services.glue.dq

El paquete com.amazonaws.services.glue.dq de la biblioteca de Scala de AWS Glue contiene las siguientes API:

- EvaluateDataQuality

com.amazonaws.services.glue.types

El paquete com.amazonaws.services.glue.types de la biblioteca AWS Glue Scala contiene las API siguientes:

- ArrayNode
- BinaryNode
- BooleanNode
- ByteNode
- DateNode
- DecimalNode
- DoubleNode
- DynamicNode
- FloatNode
- IntegerNode
- LongNode
- MapLikeNode
- MapNode
- NullNode
- ObjectNode
- ScalarNode
- ShortNode
- StringNode
- TimestampNode

com.amazonaws.services.glue.util

El paquete com.amazonaws.services.util de la biblioteca AWS Glue Scala contiene las API siguientes:

- GlueArgParser
- Trabajo

API ChoiceOption Scala de AWS Glue

Temas

- Característica ChoiceOption
- Objeto ChoiceOption
- case class ChoiceOptionWithResolver
- case class MatchCatalogSchemaChoiceOption

Paquete: com.amazonaws.services.glue

Característica ChoiceOption

```
trait ChoiceOption extends Serializable
```

Objeto ChoiceOption

ChoiceOption

```
object ChoiceOption
```

Una estrategia general para resolver la selección aplicable a todos los nodos `ChoiceType` de un objeto `DynamicFrame`.

- `val CAST`
- `val MAKE_COLS`
- `val MAKE_STRUCT`
- `val MATCH_CATALOG`

- `val PROJECT`

Def apply

```
def apply(choice: String): ChoiceOption
```

case class ChoiceOptionWithResolver

```
case class ChoiceOptionWithResolver(name: String, choiceResolver: ChoiceResolver)
 extends ChoiceOption {}
```

case class MatchCatalogSchemaChoiceOption

```
case class MatchCatalogSchemaChoiceOption() extends ChoiceOption {}
```

Abstract DataSink class

Temas

- def writeDynamicFrame
- def pyWriteDynamicFrame
- Def writeDataFrame
- Def pyWriteDataFrame
- def setCatalogInfo
- def supportsFormat
- def setFormat
- def withFormat
- def setAccumulableSize
- def getOutputErrorRecordsAccumulable
- def errorsAsDynamicFrame
- Objeto DataSink

Paquete: com.amazonaws.services.glue

```
abstract class DataSink
```

El escritor análogo a un `DataSource`. `DataSink` encapsula un destino y un formato en el que se puede escribir un objeto `DynamicFrame`.

def writeDynamicFrame

```
def writeDynamicFrame( frame : DynamicFrame,
                       callSite : CallSite = CallSite("Not provided", "")
                     ) : DynamicFrame
```

def pyWriteDynamicFrame

```
def pyWriteDynamicFrame( frame : DynamicFrame,
                         site : String = "Not provided",
                         info : String = "" )
```

Def writeDataFrame

```
def writeDataFrame(frame: DataFrame,
                   glueContext: GlueContext,
                   callSite: CallSite = CallSite("Not provided", "")
                   ): DataFrame
```

Def pyWriteDataFrame

```
def pyWriteDataFrame(frame: DataFrame,
                     glueContext: GlueContext,
                     site: String = "Not provided",
                     info: String = ""
                     ): DataFrame
```

def setCatalogInfo

```
def setCatalogInfo(catalogDatabase: String,
                   catalogTableName : String,
```

```
                    catalogId : String = "")
```

def supportsFormat

```
def supportsFormat( format : String ) : Boolean
```

def setFormat

```
def setFormat( format : String,
               options : JsonOptions
             ) : Unit
```

def withFormat

```
def withFormat( format : String,
                options : JsonOptions = JsonOptions.empty
              ) : DataSink
```

def setAccumulableSize

```
def setAccumulableSize( size : Int ) : Unit
```

def getOutputErrorRecordsAccumulable

```
def getOutputErrorRecordsAccumulable : Accumulable[List[OutputError], OutputError]
```

def errorsAsDynamicFrame

```
def errorsAsDynamicFrame : DynamicFrame
```

Objeto DataSink

```
object DataSink
```

def recordMetrics

```
def recordMetrics( frame : DynamicFrame,
                   ctxt : String
                 ) : DynamicFrame
```

Característica DataSource Scala de AWS Glue

Paquete: com.amazonaws.services.glue

Una interfaz de alto nivel para producir un `DynamicFrame`.

```
trait DataSource {

  def getDynamicFrame : DynamicFrame

  def getDynamicFrame( minPartitions : Int,
                       targetPartitions : Int
                     ) : DynamicFrame
  def getDataFrame : DataFrame

  /** @param num: the number of records for sampling.
    * @param options: optional parameters to control sampling behavior. Current
 available parameter for Amazon S3 sources in options:
    *  1. maxSamplePartitions: the maximum number of partitions the sampling will
 read.
    *  2. maxSampleFilesPerPartition: the maximum number of files the sampling will
 read in one partition.
    */
  def getSampleDynamicFrame(num:Int, options: JsonOptions = JsonOptions.empty):
 DynamicFrame

  def glueContext : GlueContext

  def setFormat( format : String,
                 options : String
               ) : Unit

  def setFormat( format : String,
                 options : JsonOptions
               ) : Unit
```

```
  def supportsFormat( format : String ) : Boolean

  def withFormat( format : String,
                  options : JsonOptions = JsonOptions.empty
                ) : DataSource
}
```

API DynamicFrame Scala de AWS Glue

Paquete: com.amazonaws.services.glue

Contenido

- Clase DynamicFrame Scala de AWS Glue
  - Val errorsCount
  - Def applyMapping
  - Def assertErrorThreshold
  - Def count
  - Def dropField
  - Def dropFields
  - Def dropNulls
  - def errorsAsDynamicFrame
  - Def filter
  - Def getName
  - Def getNumPartitions
  - Def getSchemaIfComputed
  - Def isSchemaComputed
  - Def javaToPython
  - Def join
  - Def map
  - Def mergeDynamicFrames
  - Def printSchema
  - Def recomputeSchema
  - Def relationalize
  - Def renameField

- Def repartition
- Def resolveChoice
- Def schema
- Def selectField
- Def selectFields
- Def show
- Def simplifyDDBJson
- Def spigot
- Def splitFields
- Def splitRows
- Def stageErrorsCount
- Def toDF
- Def unbox
- Def unnest
- Def unnestDDBJson
- Def withFrameSchema
- Def withName
- Def withTransformationContext

- Objeto DynamicFrame
  - Def apply
  - def emptyDynamicFrame
  - Def fromPythonRDD
  - Def ignoreErrors
  - Def inlineErrors
  - Def newFrameWithErrors

Clase DynamicFrame Scala de AWS Glue

Paquete: com.amazonaws.services.glue

```
class DynamicFrame extends Serializable with Logging  (
        val glueContext : GlueContext,
```

```
            _records : RDD[DynamicRecord],
            val name : String = s"",
            val transformationContext : String = DynamicFrame.UNDEFINED,
            callSite : CallSite = CallSite("Not provided", ""),
            stageThreshold : Long = 0,
            totalThreshold : Long = 0,
            prevErrors : => Long = 0,
            errorExpr : => Unit = {} )
```

`DynamicFrame` es una colección distribuida de objetos DynamicRecord autodescriptivos.

Los elementos `DynamicFrames` se han diseñado para proporcionar un modelo de datos flexible para operaciones de ETL (extracción, transformación y carga). No necesitan un esquema para crearse y se pueden usar para leer y transformar datos que contienen valores y tipos confusos o incoherentes. Un esquema se puede calcular bajo demanda para las operaciones que necesiten uno.

Los objetos `DynamicFrame` proporcionan una serie de transformaciones para la limpieza de datos y ETL. También admiten la conversión a clases DataFrame de SparkSQL, y desde dichas clases, para integrarse con el código existente y las numerosas operaciones de análisis que proporcionan las clases DataFrame.

Los siguientes parámetros se comparten entre las numerosas transformaciones de AWS Glue que construyen objetos `DynamicFrame`:

- `transformationContext`: identificador de este `DynamicFrame`. El parámetro `transformationContext` se utiliza como clave para el estado de marcador de flujo de trabajo que se conserva de una ejecución a otra.
- `callSite`: proporciona información de contexto para los informes de error. Estos valores se establecen automáticamente cuando se realiza la llamada desde Python.
- `stageThreshold`: número máximo de registros de error permitidos en el cálculo de este objeto `DynamicFrame` antes de generar una excepción, sin incluir los registros presentes en el objeto `DynamicFrame` anterior.
- `totalThreshold`: número máximo registros de error totales antes de que se genere una excepción, incluidos los de marcos anteriores.

Val errorsCount

```
val errorsCount
```

El número de registros de error en este objeto DynamicFrame. Se incluyen errores de las operaciones anteriores.

Def applyMapping

```
def applyMapping( mappings : Seq[Product4[String, String, String, String]],
                  caseSensitive : Boolean = true,
                  transformationContext : String = "",
                  callSite : CallSite = CallSite("Not provided", ""),
                  stageThreshold : Long = 0,
                  totalThreshold : Long = 0
                ) : DynamicFrame
```

- mappings: secuencia de mapeos para construir un objeto DynamicFrame nuevo.
- caseSensitive: indica si hay que considerar que las columnas de origen distinguen entre mayúsculas y minúsculas. Configurar este valor como falso puede ser de ayuda al realizar integraciones con almacenes que no distinguen entre mayúsculas y minúsculas, como AWS Glue Data Catalog.

Selecciona, proyecta y convierte las columnas en función de una secuencia de mapeos.

Cada mapeo se compone de una columna y un tipo de origen y una columna y un tipo de destino. Los mapeos se pueden especificar como una tupla cuádruple (source_path, source_type, target_path, target_type) o un objeto MappingSpec que contiene la misma información.

Además de usar los mapeos para proyecciones y conversiones simples, también se pueden usar para anidar o aplanar campos separando los componentes de la ruta con "." (punto).

Por ejemplo, suponga que tiene una instancia de DynamicFrame con el siguiente esquema.

```
{{{
  root
  |-- name: string
  |-- age: int
  |-- address: struct
  |    |-- state: string
  |    |-- zip: int
}}}
```

Puede hacer la siguiente llamada para quitar el anidamiento en los campos state y zip:

```
{{{
  df.applyMapping(
   Seq(("name", "string", "name", "string"),
       ("age", "int", "age", "int"),
       ("address.state", "string", "state", "string"),
       ("address.zip", "int", "zip", "int")))
}}}
```

El esquema resultante es el siguiente.

```
{{{
  root
  |-- name: string
  |-- age: int
  |-- state: string
  |-- zip: int
}}}
```

También puede utilizar `applyMapping` para volver a anidar columnas. Por ejemplo, el siguiente código invierte la transformación anterior y crea una estructura denominada `address` en el destino.

```
{{{
  df.applyMapping(
   Seq(("name", "string", "name", "string"),
       ("age", "int", "age", "int"),
       ("state", "string", "address.state", "string"),
       ("zip", "int", "address.zip", "int")))
}}}
```

Los nombres de los campos que contienen caracteres `"."` (punto) se pueden citar mediante acentos graves (` `).

> **Note**
>
> Actualmente no se puede usar el método `applyMapping` para mapear columnas anidadas en matrices.

Def assertErrorThreshold

```
def assertErrorThreshold : Unit
```

Acción que fuerza el cómputo y verifica que el número de registros de error esté por debajo de `stageThreshold` y `totalThreshold`. Genera una excepción si no se cumple ninguna de las condiciones.

Def count

```
lazy
def count
```

Devuelve el número de elementos en este objeto `DynamicFrame`.

Def dropField

```
def dropField( path : String,
               transformationContext : String = "",
               callSite : CallSite = CallSite("Not provided", ""),
               stageThreshold : Long = 0,
               totalThreshold : Long = 0
             ) : DynamicFrame
```

Devuelve un objeto `DynamicFrame` nuevo sin la columna especificada.

Def dropFields

```
def dropFields( fieldNames : Seq[String],   // The column names to drop.
                transformationContext : String = "",
                callSite : CallSite = CallSite("Not provided", ""),
                stageThreshold : Long = 0,
                totalThreshold : Long = 0
              ) : DynamicFrame
```

Devuelve un objeto `DynamicFrame` nuevo sin las columnas especificadas.

Este método se puede utilizar para eliminar columnas anidadas, incluidas las que se encuentran en matrices, pero no para descartar elementos de matriz específicos.

Def dropNulls

```
def dropNulls( transformationContext : String = "",
               callSite : CallSite = CallSite("Not provided", ""),
               stageThreshold : Long = 0,
               totalThreshold : Long = 0 )
```

Devuelve un objeto `DynamicFrame` nuevo sin las columnas nulas.

> **Note**
>
> Solo se eliminan las columnas del tipo `NullType`. Los valores nulos individuales de otras columnas no se eliminan ni modifican.

def errorsAsDynamicFrame

```
def errorsAsDynamicFrame
```

Devuelve un objeto `DynamicFrame` nuevo que contiene los registros de error de este `DynamicFrame`.

Def filter

```
def filter( f : DynamicRecord => Boolean,
            errorMsg : String = "",
            transformationContext : String = "",
            callSite : CallSite = CallSite("Not provided"),
            stageThreshold : Long = 0,
            totalThreshold : Long = 0
          ) : DynamicFrame
```

Crea un objeto `DynamicFrame` nuevo que solo contiene los registros en los que la función "f" devuelve `true`. La función de filtro "f" no debe mutar el registro de entrada.

Def getName

```
def getName : String
```

Devuelve el nombre de este objeto `DynamicFrame`.

Def getNumPartitions

```
def getNumPartitions
```

Devuelve el número de particiones en este objeto `DynamicFrame`.

Def getSchemaIfComputed

```
def getSchemaIfComputed : Option[Schema]
```

Devuelve el esquema si ya se ha calculado. No analiza los datos si aún no se ha calculado el esquema.

Def isSchemaComputed

```
def isSchemaComputed : Boolean
```

Devuelve `true` si el esquema se ha calculado para este objeto `DynamicFrame`, de lo contrario devuelve `false`. Si este método devuelve false, la llamada al método `schema` necesitará otra pasada por los registros en este objeto `DynamicFrame`.

Def javaToPython

```
def javaToPython : JavaRDD[Array[Byte]]
```

Def join

```
def join( keys1 : Seq[String],
          keys2 : Seq[String],
          frame2 : DynamicFrame,
          transformationContext : String = "",
          callSite : CallSite = CallSite("Not provided", ""),
          stageThreshold : Long = 0,
          totalThreshold : Long = 0
        ) : DynamicFrame
```

- `keys1`: las columnas de este objeto `DynamicFrame` que se utilizarán para la combinación.

- `keys2`: las columnas de `frame2` que se utilizarán para la combinación. Deben tener la misma longitud que `keys1`.
- `frame2`: el elemento `DynamicFrame` con el que debe realizarse la combinación.

Devuelve el resultado de realizar una combinación equijoin con `frame2` utilizando las claves especificadas.

Def map

```
def map( f : DynamicRecord => DynamicRecord,
         errorMsg : String = "",
         transformationContext : String = "",
         callSite : CallSite = CallSite("Not provided", ""),
         stageThreshold : Long = 0,
         totalThreshold : Long = 0
       ) : DynamicFrame
```

Devuelve un objeto `DynamicFrame` nuevo creado aplicando la función "f" especificada a cada registro de este objeto `DynamicFrame`.

Este método copia cada registro antes de aplicar la función especificada, por lo que es seguro para mutar los registros. Si la función de mapeo genera una excepción en un registro determinado, dicho registro se marca como un error y la pila de seguimiento se guarda como una columna en el registro de errores.

Def mergeDynamicFrames

```
def mergeDynamicFrames( stageDynamicFrame: DynamicFrame,  primaryKeys: Seq[String],
 transformationContext: String = "",
                         options: JsonOptions = JsonOptions.empty, callSite: CallSite =
 CallSite("Not provided"),
                         stageThreshold: Long = 0, totalThreshold: Long = 0):
 DynamicFrame
```

- `stageDynamicFrame`: `DynamicFrame` provisional que se fusionará.
- `primaryKeys`: la lista de campos de clave principal para hacer coincidir los registros de los `DynamicFrame` de origen y provisionales.
- `transformationContext`: una cadena única que se utiliza para recuperar metadatos sobre la transformación actual (opcional).

- `options`: una cadena de pares de nombre-valor de JSON que proporcionan información adicional para esta transformación.
- `callSite`: se utiliza para proporcionar información de contexto para los informes de error.
- `stageThreshold`: un valor `Long`. Número de errores de la transformación especificada que provocarán que el proceso se termine.
- `totalThreshold`: un valor `Long`. Número total de errores hasta esta transformación (incluida) que provocarán que el proceso se termine.

Combina este objeto `DynamicFrame` con una instancia provisional de `DynamicFrame` en función de las claves principales especificadas para identificar registros. Los registros duplicados (registros con las mismas claves principales) no se eliminan. Si no hay ningún registro que coincida en el marco provisional, se retienen todos los registros del origen (incluidos los duplicados). Si el marco provisional tiene registros coincidentes, estos sobrescriben a los registros del origen en AWS Glue.

La instancia de `DynamicFrame` devuelta contiene el registro A en los siguientes casos:

1. Si `A` existe tanto en el marco de origen como en el marco provisional, se devuelve `A` en el marco provisional.
2. Si `A` está en la tabla de origen y `A.primaryKeys` no está en `stagingDynamicFrame` (significa que `A` no se actualiza en la tabla provisional).

No es necesario que el marco de origen y el marco provisional tengan el mismo esquema.

Example

```
val mergedFrame: DynamicFrame = srcFrame.mergeDynamicFrames(stageFrame, Seq("id1",
 "id2"))
```

Def printSchema

```
def printSchema : Unit
```

Imprime el esquema de este objeto `DynamicFrame` en `stdout` en un formato fácil de leer.

Def recomputeSchema

```
def recomputeSchema : Schema
```

Fuerza un nuevo cálculo del esquema. Es necesario realizar un análisis de los datos, lo que podría "endurecer" el esquema si hay algunos campos del esquema actual que no están en los datos.

Devuelve el esquema calculado de nuevo.

Def relationalize

```
def relationalize( rootTableName : String,
                   stagingPath : String,
                   options : JsonOptions = JsonOptions.empty,
                   transformationContext : String = "",
                   callSite : CallSite = CallSite("Not provided"),
                   stageThreshold : Long = 0,
                   totalThreshold : Long = 0
                 ) : Seq[DynamicFrame]
```

- `rootTableName`: el nombre que se va a utilizar para el `DynamicFrame` base en la salida. Los `DynamicFrame` que se crean mediante matrices dinámicas comienzan con este prefijo.
- `stagingPath`: la ruta de Amazon Simple Storage Service (Amazon S3) para escribir datos intermedios.
- `options`: opciones y configuración de Rationalize. En la actualidad no se utiliza.

Aplana todas las estructuras anidadas y convierte las matrices en tablas independientes.

Puede utilizar esta operación para preparar datos con anidamiento profundo para su incorporación a una base de datos relacional. Las estructuras anidadas se aplanan de la misma manera en que lo haría la transformación Unnest. Además, las matrices se dividen en tablas independientes en las que cada elemento de matriz se convierte en una fila. Por ejemplo, supongamos que tiene una instancia de `DynamicFrame` con los siguientes datos.

```
 {"name": "Nancy", "age": 47, "friends": ["Fred", "Lakshmi"]}
 {"name": "Stephanie", "age": 28, "friends": ["Yao", "Phil", "Alvin"]}
 {"name": "Nathan", "age": 54, "friends": ["Nicolai", "Karen"]}
```

Ejecute el siguiente código.

```
{{{
  df.relationalize("people", "s3:/my_bucket/my_path", JsonOptions.empty)
}}}
```

Se generan dos tablas. La primera tabla se denomina "people" y contiene lo siguiente.

```
{{{
  {"name": "Nancy", "age": 47, "friends": 1}
  {"name": "Stephanie", "age": 28, "friends": 2}
  {"name": "Nathan", "age": 54, "friends": 3)
}}}
```

Aquí, la matriz friends se ha reemplazado por una clave de combinación generada automáticamente. Se crea una tabla diferente llamada `people.friends` con el siguiente contenido.

```
{{{
  {"id": 1, "index": 0, "val": "Fred"}
  {"id": 1, "index": 1, "val": "Lakshmi"}
  {"id": 2, "index": 0, "val": "Yao"}
  {"id": 2, "index": 1, "val": "Phil"}
  {"id": 2, "index": 2, "val": "Alvin"}
  {"id": 3, "index": 0, "val": "Nicolai"}
  {"id": 3, "index": 1, "val": "Karen"}
}}}
```

En esta tabla, "`id`" es una clave de combinación que identifica de qué registro procede el elemento de matriz, "`index`" hace referencia a la posición en la matriz original y "`val`" es la entrada de matriz real.

El método `relationalize` devuelve la secuencia de objetos `DynamicFrame` creados aplicando este proceso de forma recursiva a todas las matrices.

> **Note**
>
> La biblioteca de AWS Glue genera automáticamente las claves de combinación de las tablas nuevas. Para garantizar que las claves de combinación sean únicas en las ejecuciones de flujo de trabajo, debe habilitar los marcadores de flujo de trabajo.

Def renameField

```
def renameField( oldName : String,
                 newName : String,
                 transformationContext : String = "",
```

```
                callSite : CallSite = CallSite("Not provided", ""),
                stageThreshold : Long = 0,
                totalThreshold : Long = 0
              ) : DynamicFrame
```

- `oldName`: el nombre original de la columna.
- `newName`: el nuevo nombre de la columna.

Devuelve un `DynamicFrame` nuevo que incluye el campo especificado con un nuevo nombre.

Puede usar este método para cambiar el nombre de los campos anidados. Por ejemplo, el siguiente código cambiaría el nombre de `state` a `state_code` en la estructura de dirección.

```
{{{
  df.renameField("address.state", "address.state_code")
}}}
```

Def repartition

```
def repartition( numPartitions : Int,
                transformationContext : String = "",
                callSite : CallSite = CallSite("Not provided", ""),
                stageThreshold : Long = 0,
                totalThreshold : Long = 0
              ) : DynamicFrame
```

Devuelve un objeto `DynamicFrame` nuevo con `numPartitions` particiones.

Def resolveChoice

```
def resolveChoice( specs : Seq[Product2[String, String]] = Seq.empty[ResolveSpec],
                   choiceOption : Option[ChoiceOption] = None,
                   database : Option[String] = None,
                   tableName : Option[String] = None,
                   transformationContext : String = "",
                   callSite : CallSite = CallSite("Not provided", ""),
                   stageThreshold : Long = 0,
                   totalThreshold : Long = 0
                 ) : DynamicFrame
```

- `choiceOption`: acción que se aplica a todas las columnas `ChoiceType` que no están indicadas en la secuencia specs.
- `database`: la base de datos de Data Catalog que se usará con la acción `match_catalog`.
- `tableName`: la tabla de Data Catalog que se usará con la acción `match_catalog`.

Devuelve un objeto `DynamicFrame` nuevo reemplazando uno o varios `ChoiceType` por un tipo más específico.

Hay dos formas de usar `resolveChoice`. La primera consiste en especificar una secuencia de columnas específicas y cómo resolverlas. Se especifican como tuplas compuestas de pares (columna, acción).

Los posibles valores son los siguientes:

- `cast:type`: intenta convertir todos los valores al tipo especificado.
- `make_cols`: convierte cada tipo diferenciado en una columna con el nombre `columnName_type`.
- `make_struct`: convierte una columna en una estructura con claves para cada tipo diferenciado.
- `project:type`: retiene solo valores del tipo especificado.

El otro modo de `resolveChoice` es especificar una resolución única para todas las columnas `ChoiceType`. Se puede utilizar en los casos en los que se desconozca la lista completa de `ChoiceType` antes de la ejecución. Además de las acciones enumeradas anteriormente, este modo también admite la siguiente acción:

- `match_catalog`: intenta convertir cada `ChoiceType` al tipo correspondiente en la tabla de catálogos especificada.

Ejemplos:

Resolver la columna `user.id` mediante la conversión a int y hacer que el campo `address` solo conserve estructuras.

```
{{{
  df.resolveChoice(specs = Seq(("user.id", "cast:int"), ("address", "project:struct")))
}}}
```

Resolver todas las columnas `ChoiceType` convirtiendo cada opción en una columna independiente.

```
{{{
  df.resolveChoice(choiceOption = Some(ChoiceOption("make_cols")))
}}}
```

Resolver todas las columnas `ChoiceType` mediante la conversión a los tipos de la tabla de catálogos especificada.

```
{{{
  df.resolveChoice(choiceOption = Some(ChoiceOption("match_catalog")),
                   database = Some("my_database"),
                   tableName = Some("my_table"))
}}}
```

Def schema

```
def schema : Schema
```

Devuelve el esquema de este objeto `DynamicFrame`.

El esquema devuelto contiene todos los campos presentes en un registro en este objeto `DynamicFrame`. Sin embargo, en un pequeño número de casos, es posible que también contenga campos adicionales. Puede usar el método Unnest para "ajustar" el esquema en función de los registros de este objeto `DynamicFrame`.

Def selectField

```
def selectField( fieldName : String,
                 transformationContext : String = "",
                 callSite : CallSite = CallSite("Not provided", ""),
                 stageThreshold : Long = 0,
                 totalThreshold : Long = 0
               ) : DynamicFrame
```

Devuelve un campo individual como un objeto `DynamicFrame`.

Def selectFields

```
def selectFields( paths : Seq[String],
                  transformationContext : String = "",
```

```
                callSite : CallSite = CallSite("Not provided", ""),
                stageThreshold : Long = 0,
                totalThreshold : Long = 0
              ) : DynamicFrame
```

- `paths`: secuencia de los nombres de columna que se seleccionarán.

Devuelve un nuevo objeto `DynamicFrame` que contiene las columnas especificadas.

> **Note**
>
> Solo puede usar el método `selectFields` para seleccionar columnas de nivel superior. Puede usar el método applyMapping para seleccionar columnas anidadas.

Def show

```
def show( numRows : Int = 20 ) : Unit
```

- `numRows`: número de filas que se imprimirán.

Imprime filas de este objeto `DynamicFrame` en formato JSON.

Def simplifyDDBJson

Las exportaciones de DynamoDB con el conector de exportaciones de DynamoDB de AWS Glue generan archivos JSON con estructuras anidadas específicas. Para obtener más información, consulte Objetos de datos. `simplifyDDBJson` Simplifica las columnas anidadas en un DynamicFrame de este tipo de datos y genera un DynamicFrame nuevo y simplificado. Si hay varios tipos y un tipo de mapa contenidos en tipo de lista, los elementos en la lista no se simplificarán. Este método solo es compatible con los datos exportados de DynamoDB con el formato JSON. Considere la acción de `unnest` para realizar cambios similares en otros tipos de datos.

```
def simplifyDDBJson() : DynamicFrame
```

Este método no toma ningún parámetro.

Ejemplo de entrada

Considere el siguiente esquema generado por una exportación a DynamoDB:

```
root
|-- Item: struct
|    |-- parentMap: struct
|    |    |-- M: struct
|    |    |    |-- childMap: struct
|    |    |    |    |-- M: struct
|    |    |    |    |    |-- appName: struct
|    |    |    |    |    |    |-- S: string
|    |    |    |    |    |-- packageName: struct
|    |    |    |    |    |    |-- S: string
|    |    |    |    |    |-- updatedAt: struct
|    |    |    |    |    |    |-- N: string
|    |-- strings: struct
|    |    |-- SS: array
|    |    |    |-- element: string
|    |-- numbers: struct
|    |    |-- NS: array
|    |    |    |-- element: string
|    |-- binaries: struct
|    |    |-- BS: array
|    |    |    |-- element: string
|    |-- isDDBJson: struct
|    |    |-- BOOL: boolean
|    |-- nullValue: struct
|    |    |-- NULL: boolean
```

Código de ejemplo

```
import com.amazonaws.services.glue.GlueContext
import com.amazonaws.services.glue.util.GlueArgParser
import com.amazonaws.services.glue.util.Job
import com.amazonaws.services.glue.util.JsonOptions
import com.amazonaws.services.glue.DynamoDbDataSink
import org.apache.spark.SparkContextimport scala.collection.JavaConverters._

object GlueApp {

  def main(sysArgs: Array[String]): Unit = {
    val glueContext = new GlueContext(SparkContext.getOrCreate())
    val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB_NAME").toArray)
    Job.init(args("JOB_NAME"), glueContext, args.asJava)
```

```
    val dynamicFrame = glueContext.getSourceWithFormat(
      connectionType = "dynamodb",
      options = JsonOptions(Map(
        "dynamodb.export" -> "ddb",
        "dynamodb.tableArn" -> "ddbTableARN",
        "dynamodb.s3.bucket" -> "exportBucketLocation",
        "dynamodb.s3.prefix" -> "exportBucketPrefix",
        "dynamodb.s3.bucketOwner" -> "exportBucketAccountID",
      ))
    ).getDynamicFrame()

    val simplified = dynamicFrame.simplifyDDBJson()
    simplified.printSchema()

    Job.commit()
  }

}
```

Ejemplo de resultado

La transformación de `simplifyDDBJson` simplificará esto de la siguiente forma:

```
root
|-- parentMap: struct
|    |-- childMap: struct
|    |    |-- appName: string
|    |    |-- packageName: string
|    |    |-- updatedAt: string
|-- strings: array
|    |-- element: string
|-- numbers: array
|    |-- element: string
|-- binaries: array
|    |-- element: string
|-- isDDBJson: boolean
|-- nullValue: null
```

Def spigot

```
def spigot( path : String,
            options : JsonOptions = new JsonOptions("{}"),
```

```
            transformationContext : String = "",
            callSite : CallSite = CallSite("Not provided"),
            stageThreshold : Long = 0,
            totalThreshold : Long = 0
          ) : DynamicFrame
```

Transformación de paso a través que devuelve los mismos registros, pero escribe un subconjunto de registros como efecto secundario.

- `path`: ruta de Amazon S3 en la que se escribirá la salida, con el formato `s3://bucket//path`.
- `options`: mapa `JsonOptions` opcional que describe el comportamiento de muestreo.

Devuelve un objeto `DynamicFrame` que contiene los mismos registros que este.

De forma predeterminada, escribe 100 registros arbitrarios en la ubicación especificada por `path`. Puede personalizar este comportamiento mediante el mapa `options`. Entre las claves válidas se incluyen:

- `topk`: especifica el número total de registros escritos. El valor predeterminado es 100.
- `prob`: especifica la probabilidad de que se incluya un registro individual (como decimal). El valor predeterminado es 1.

Por ejemplo, la siguiente llamada realizaría un muestreo del conjunto de datos seleccionando cada registro con una probabilidad del 20 % y se detendría después de que se hubieran escrito 200 registros.

```
{{{
  df.spigot("s3://my_bucket/my_path", JsonOptions(Map("topk" -&gt; 200, "prob" -&gt;
 0.2)))
}}}
```

Def splitFields

```
def splitFields( paths : Seq[String],
                 transformationContext : String = "",
                 callSite : CallSite = CallSite("Not provided", ""),
                 stageThreshold : Long = 0,
                 totalThreshold : Long = 0
               ) : Seq[DynamicFrame]
```

- `paths`: las rutas que se incluirán en el primer objeto `DynamicFrame`.

Devuelve una secuencia de dos objetos `DynamicFrame`. El primer `DynamicFrame` contiene las rutas especificadas y el segundo contiene las demás columnas.

Ejemplo

En este ejemplo, se toma un objeto DynamicFrame creado a partir de la tabla `persons` en la base de datos `legislators` de Data Catalog de AWS Glue y divide el DynamicFrame en dos, con los campos especificados incluidos en el primer DynamicFrame y los campos restantes incluidos en un segundo DynamicFrame. Luego, el ejemplo elige el primer DynamicFrame del resultado.

```
val InputFrame = glueContext.getCatalogSource(database="legislators",
 tableName="persons",
transformationContext="InputFrame").getDynamicFrame()

val SplitField_collection = InputFrame.splitFields(paths=Seq("family_name", "name",
 "links.note",
"links.url", "gender", "image", "identifiers.scheme", "identifiers.identifier",
 "other_names.lang",
"other_names.note", "other_names.name"), transformationContext="SplitField_collection")

val ResultFrame = SplitField_collection(0)
```

Def splitRows

```
def splitRows( paths : Seq[String],
               values : Seq[Any],
               operators : Seq[String],
               transformationContext : String,
               callSite : CallSite,
               stageThreshold : Long,
               totalThreshold : Long
             ) : Seq[DynamicFrame]
```

Divide las filas en función de predicados que comparan columnas con constantes.

- `paths`: columnas que se van a utilizar para la comparación.
- `values`: valores constantes que se van a utilizar para la comparación.
- `operators`: operadores que se van a utilizar para la comparación.

Devuelve una secuencia de dos objetos DynamicFrame. El primero contiene filas en las que se aplica el predicado y el segundo contiene las que no se les aplica.

Los predicados se especifican usando tres secuencias: "paths" contiene los nombres de columna (posiblemente anidadas), "values" contiene los valores constantes con los que se realizará la comparación y "operators" contiene los operadores que se usarán para la comparación. Las tres secuencias deben tener la misma longitud: el operador que ocupa la posición n se usa para comparar la columna n con el valor n.

Cada operador debe ser uno de los siguientes: "!=", "=", "<=", "<", ">=" o ">".

A modo de ejemplo, la siguiente llamada dividiría un objeto DynamicFrame de modo que el primer marco de salida contuviera los registros de personas de más de 65 años de Estados Unidos y el segundo contuviera todos los demás registros.

```
{{{
  df.splitRows(Seq("age", "address.country"), Seq(65, "USA"), Seq("&gt;=", "="))
}}}
```

Def stageErrorsCount

```
def stageErrorsCount
```

Devuelve el número de registros de error creado al calcular este objeto DynamicFrame. Se excluyen los errores de operaciones anteriores que se pasaron a este objeto DynamicFrame como entrada.

Def toDF

```
def toDF( specs : Seq[ResolveSpec] = Seq.empty[ResolveSpec] ) : DataFrame
```

Convierte este DynamicFrame en un DataFrame Apache Spark SQL con el mismo esquema y registros.

> **Note**
>
> Dado que los objetos DataFrame no admiten ChoiceType, este método convierte automáticamente las columnas ChoiceType en StructType. Para obtener más información y opciones para la resolución de elección, consulte resolveChoice.

Def unbox

```
def unbox( path : String,
           format : String,
           optionString : String = "{}",
           transformationContext : String = "",
           callSite : CallSite = CallSite("Not provided"),
           stageThreshold : Long = 0,
           totalThreshold : Long = 0
         ) : DynamicFrame
```

- `path`: la columna que se va a analizar. Debe tener formato de cadena o binario.
- `format`: el formato que se utilizará para el análisis.
- `optionString`: opciones que se pasarán al formato, como el separador CSV.

Analiza una cadena insertada o una columna binaria de acuerdo con el formato especificado. Las columnas analizadas se anidan en una estructura con el nombre de columna original.

Por ejemplo, suponga que tiene un archivo CSV con una columna JSON insertada.

```
name, age, address
Sally, 36, {"state": "NE", "city": "Omaha"}
...
```

Después de un análisis inicial, obtendrá un objeto `DynamicFrame` con el siguiente esquema.

```
{{{
  root
  |-- name: string
  |-- age: int
  |-- address: string
}}}
```

Puede llamar a `unbox` en la columna de dirección para analizar los componentes específicos.

```
{{{
  df.unbox("address", "json")
}}}
```

Esto nos proporciona un objeto `DynamicFrame` con el siguiente esquema.

```
{{{
  root
  |-- name: string
  |-- age: int
  |-- address: struct
  |    |-- state: string
  |    |-- city: string
}}}
```

Def unnest

```
def unnest( transformationContext : String = "",
            callSite : CallSite = CallSite("Not Provided"),
            stageThreshold : Long = 0,
            totalThreshold : Long = 0
          ) : DynamicFrame
```

Devuelve un objeto `DynamicFrame` nuevo con todas las estructuras anidadas aplanadas. Los nombres se crean con el carácter `"."` (punto).

Por ejemplo, suponga que tiene una instancia de `DynamicFrame` con el siguiente esquema.

```
{{{
  root
  |-- name: string
  |-- age: int
  |-- address: struct
  |    |-- state: string
  |    |-- city: string
}}}
```

La siguiente llamada elimina el anidamiento de la estructura de dirección:

```
{{{
  df.unnest()
}}}
```

El esquema resultante es el siguiente.

```
{{{
  root
  |-- name: string
  |-- age: int
  |-- address.state: string
  |-- address.city: string
}}}
```

Este método también aplana las estructuras anidadas de dentro de las matrices. Sin embargo, por razones históricas, los nombres de dichos campos tienen anexado previamente el nombre de la matriz contenedora y "`.val`".

Def unnestDDBJson

```
unnestDDBJson(transformationContext : String = "",
            callSite : CallSite = CallSite("Not Provided"),
            stageThreshold : Long = 0,
            totalThreshold : Long = 0): DynamicFrame
```

Desanida las columnas anidadas de un elemento `DynamicFrame` que se encuentren específicamente en la estructura JSON de DynamoDB y devuelve un nuevo elemento `DynamicFrame` no anidado. Las columnas que sean de una matriz de tipos de estructuras no se desanidarán. Tenga en cuenta que esta transformación de desanidamiento es un tipo específico que se comporta de modo diferente a la transformación `unnest` normal y requiere que los datos ya estén en la estructura JSON de DynamoDB. Para obtener más información, consulte JSON de DynamoDB.

Por ejemplo, el esquema de una lectura de una exportación con la estructura JSON de DynamoDB puede parecerse al siguiente:

```
root
|-- Item: struct
|    |-- ColA: struct
|    |    |-- S: string
|    |-- ColB: struct
|    |    |-- S: string
|    |-- ColC: struct
|    |    |-- N: string
|    |-- ColD: struct
|    |    |-- L: array
|    |    |    |-- element: null
```

La transformación `unnestDDBJson()` convertiría esto en:

```
root
|-- ColA: string
|-- ColB: string
|-- ColC: string
|-- ColD: array
|    |-- element: null
```

En el siguiente ejemplo de código, se muestra cómo utilizar el conector de exportación de DynamoDB de AWS Glue, invocar un desanidamiento JSON de DynamoDB e imprimir el número de particiones:

```
import com.amazonaws.services.glue.GlueContext
import com.amazonaws.services.glue.util.GlueArgParser
import com.amazonaws.services.glue.util.Job
import com.amazonaws.services.glue.util.JsonOptions
import com.amazonaws.services.glue.DynamoDbDataSink
import org.apache.spark.SparkContext
import scala.collection.JavaConverters._


object GlueApp {

  def main(sysArgs: Array[String]): Unit = {
    val glueContext = new GlueContext(SparkContext.getOrCreate())
    val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB_NAME").toArray)
    Job.init(args("JOB_NAME"), glueContext, args.asJava)

    val dynamicFrame = glueContext.getSourceWithFormat(
      connectionType = "dynamodb",
      options = JsonOptions(Map(
        "dynamodb.export" -> "ddb",
        "dynamodb.tableArn" -> "<test_source>",
        "dynamodb.s3.bucket" -> "<bucket name>",
        "dynamodb.s3.prefix" -> "<bucket prefix>",
        "dynamodb.s3.bucketOwner" -> "<account_id of bucket>",
      ))
    ).getDynamicFrame()

    val unnested = dynamicFrame.unnestDDBJson()
    print(unnested.getNumPartitions())
```

```
    Job.commit()
  }

}
```

Def withFrameSchema

```
def withFrameSchema( getSchema : () => Schema ) : DynamicFrame
```

- `getSchema`: función que devuelve el esquema que se va a utilizar. Se especifica como una función sin parámetros para diferir el cómputo potencialmente costoso.

Establece el esquema de este objeto `DynamicFrame` en el valor especificado. Se utiliza principalmente de forma interna para evitar un nuevo y costoso cálculo de esquema. El esquema pasado debe contener todas las columnas presentes en los datos.

Def withName

```
def withName( name : String ) : DynamicFrame
```

- `name`: nuevo nombre que se usará.

Devuelve una copia de este objeto `DynamicFrame` con un nuevo nombre.

Def withTransformationContext

```
def withTransformationContext( ctx : String ) : DynamicFrame
```

Devuelve una copia de este objeto `DynamicFrame` con el contexto de transformación especificado.

Objeto DynamicFrame

Paquete: com.amazonaws.services.glue

```
object DynamicFrame
```

Def apply

```
def apply( df : DataFrame,
```

```
            glueContext : GlueContext
          ) : DynamicFrame
```

def emptyDynamicFrame

```
def emptyDynamicFrame( glueContext : GlueContext ) : DynamicFrame
```

Def fromPythonRDD

```
def fromPythonRDD( rdd : JavaRDD[Array[Byte]],
                   glueContext : GlueContext
                 ) : DynamicFrame
```

Def ignoreErrors

```
def ignoreErrors( fn : DynamicRecord => DynamicRecord ) : DynamicRecord
```

Def inlineErrors

```
def inlineErrors( msg : String,
                  callSite : CallSite
                ) : (DynamicRecord => DynamicRecord)
```

Def newFrameWithErrors

```
def newFrameWithErrors( prevFrame : DynamicFrame,
                        rdd : RDD[DynamicRecord],
                        name : String = "",
                        transformationContext : String = "",
                        callSite : CallSite,
                        stageThreshold : Long,
                        totalThreshold : Long
                      ) : DynamicFrame
```

Clase DynamicRecord Scala de AWS Glue

Temas

- Def addField
- Def dropField
- Def setError
- Def isError
- Def getError
- Def clearError
- Def write
- Def readFields
- Def clone
- Def schema
- Def getRoot
- Def toJson
- Def getFieldNode
- Def getField
- Def hashCode
- Def equals
- Objeto DynamicRecord
- Característica RecordTraverser

Paquete: com.amazonaws.services.glue

```
class DynamicRecord extends Serializable with Writable with Cloneable
```

Un objeto `DynamicRecord` es una estructura de datos autodescriptiva que representa una fila de datos en el conjunto de datos que se está procesando. Es autodescriptiva en el sentido de que se puede obtener el esquema de la fila representado mediante el objeto `DynamicRecord` si se inspecciona el registro. Un objeto `DynamicRecord` es similar a `Row` en Apache Spark.

Def addField

```
def addField( path : String,
              dynamicNode : DynamicNode
```

```
            ) : Unit
```

Agrega un objeto DynamicNode a la ruta especificada.

- `path`: ruta del campo que se va a agregar.
- `dynamicNode`: DynamicNode que se agregará a la ruta especificada.

Def dropField

```
def dropField(path: String, underRename: Boolean = false): Option[DynamicNode]
```

Descarta un objeto DynamicNode de la ruta especifica y devuelve el nodo descartado si no hay una matriz en la ruta especificada.

- `path`: ruta al campo que se descartará.
- `underRename`: true si se llama a `dropField` como parte de una transformación de cambio de nombre; de lo contrario, false (false de forma predeterminada).

Devuelve `scala.Option Option` (DynamicNode).

Def setError

```
def setError( error : Error )
```

Establece este registro como un registro de error, tal y como se especifica mediante el parámetro `error`.

Devuelve `DynamicRecord`.

Def isError

```
def isError
```

Comprueba si es un registro de error.

Def getError

```
def getError
```

Obtiene el `Error` si es un registro de error. Devuelve `scala.Some` Some (Error) si es un registro de error; de lo contrario, `scala.None`.

Def clearError

```
def clearError
```

Configure el `Error` en `scala.None.None`.

Def write

```
override def write( out : DataOutput ) : Unit
```

Def readFields

```
override def readFields( in : DataInput ) : Unit
```

Def clone

```
override def clone : DynamicRecord
```

Clona este registro en un nuevo objeto `DynamicRecord` y lo devuelve.

Def schema

```
def schema
```

Obtiene el objeto `Schema` mediante la inspección del registro.

Def getRoot

```
def getRoot : ObjectNode
```

Obtiene el objeto `ObjectNode` raíz del registro.

Def toJson

```
def toJson : String
```

Obtiene la cadena JSON del registro.

Def getFieldNode

```
def getFieldNode( path : String ) : Option[DynamicNode]
```

Obtiene el valor del campo en la `path` especificada como una opción de `DynamicNode`.

Devuelve `scala.Some` Some (DynamicNode) si el campo existe; de lo contrario, `scala.None.None`.

Def getField

```
def getField( path : String ) : Option[Any]
```

Obtiene el valor del campo en la `path` especificada como una opción de `DynamicNode`.

Devuelve `scala.Some` Some (valor).

Def hashCode

```
override def hashCode : Int
```

Def equals

```
override def equals( other : Any )
```

Objeto DynamicRecord

```
object DynamicRecord
```

Def apply

```
def apply( row : Row,
           schema : SparkStructType )
```

Aplique el método para convertir un objeto Row de Apache Spark SQL en un objeto DynamicRecord.

- `row`: un objeto Row de Spark SQL.
- `schema`: el objeto Schema de la fila.

Devuelve `DynamicRecord`.

Característica RecordTraverser

```
trait RecordTraverser {
  def nullValue(): Unit
  def byteValue(value: Byte): Unit
  def binaryValue(value: Array[Byte]): Unit
  def booleanValue(value: Boolean): Unit
  def shortValue(value: Short) : Unit
  def intValue(value: Int) : Unit
  def longValue(value: Long) : Unit
  def floatValue(value: Float): Unit
  def doubleValue(value: Double): Unit
  def decimalValue(value: BigDecimal): Unit
  def stringValue(value: String): Unit
  def dateValue(value: Date): Unit
  def timestampValue(value: Timestamp): Unit
  def objectStart(length: Int): Unit
  def objectKey(key: String): Unit
  def objectEnd(): Unit
  def mapStart(length: Int): Unit
  def mapKey(key: String): Unit
  def mapEnd(): Unit
  def arrayStart(length: Int): Unit
  def arrayEnd(): Unit
}
```

AWS GlueAPI de Scala GlueContext

Paquete: com.amazonaws.services.glue

```
class GlueContext extends SQLContext(sc) (
           @transient val sc : SparkContext,
           val defaultSourcePartitioner : PartitioningStrategy )
```

`GlueContext` es el punto de entrada para leer y escribir un DynamicFrame desde y hacia Amazon Simple Storage Service (Amazon S3), el AWS Glue Data Catalog, JDBC, etc. Esta clase ofrece

funciones de utilidades para crear objetos Característica DataSource y DataSink que, a su vez, se pueden usar para leer y escribir objetos `DynamicFrame`.

También se puede utilizar `GlueContext` para establecer un número de particiones de destino (el valor predeterminado es 20) en el objeto `DynamicFrame` si el número de particiones creadas desde el origen es menor que un umbral mínimo para las particiones (el valor predeterminado es 10).

columnas def addIngestionTime

```
def addIngestionTimeColumns(
         df : DataFrame,
         timeGranularity : String = "") : dataFrame
```

Agrega columnas de tiempo de ingesta como `ingest_year`, `ingest_month`, `ingest_day`, `ingest_hour`, `ingest_minute` al `DataFrame` de entrada. Esta función se genera en forma automática en el script generado por AWS Glue cuando especifique una tabla del Catálogo de datos con Amazon S3 como destino. Esta función actualiza en forma automática la partición con columnas de tiempo de ingesta en la tabla de salida. Esto permite que los datos de salida se dividan automáticamente en el tiempo de ingesta sin requerir columnas de tiempo de ingesta explícitas en los datos de entrada.

- `dataFrame`: el `dataFrame` al que anexar las columnas de tiempo de ingesta.
- `timeGranularity`: la granularidad de las columnas de tiempo. Los valores válidos son “`day`”, “`hour`” y “`minute`”. Por ejemplo, si “`hour`” se transfiere a la función, el `dataFrame` original tendrá las columnas de tiempo “`ingest_year`,” “`ingest_month`,” “`ingest_day`” y “`ingest_hour`” anexadas.

Devuelve el marco de datos después de anexar las columnas de granularidad de tiempo.

Ejemplo:

```
glueContext.addIngestionTimeColumns(dataFrame, "hour")
```

def createDataFrame FromOptions

```
def createDataFrameFromOptions( connectionType : String,
                        connectionOptions : JsonOptions,
                        transformationContext : String = "",
```

```
                        format : String = null,
                        formatOptions : JsonOptions = JsonOptions.empty
                      ) : DataSource
```

Muestra un `DataFrame` que se crea con la conexión y el formato especificados. Utilice esta función únicamente con fuentes de streaming de AWS Glue.

- `connectionType`: el tipo de conexión de streaming. Los valores válidos son `kinesis` y `kafka`.
- `connectionOptions`: opciones de conexión, que son diferentes para Kinesis y Kafka. Puede encontrar la lista de todas las opciones de conexión para cada origen de datos de streaming en Tipos de conexión y opciones para ETL en AWS Glue para Spark. Tenga en cuenta las siguientes diferencias en las opciones de conexión de streaming:
  - Los orígenes de streaming de Kinesis requieren `streamARN`, `startingPosition`, `inferSchema` y `classification`.
  - Los orígenes de streaming de Kafka requieren `connectionName`, `topicName`, `startingOffsets`, `inferSchema` y `classification`.
- `transformationContext`: el contexto de transformación que se va a utilizar (opcional).
- `format`: una especificación de formato (opcional). Se utiliza con una conexión de Amazon S3 o AWS Glue que admite diversos formatos. Para obtener información acerca de los formatos soportados, consulte Opciones de formato de datos para las entradas y las salidas en AWS Glue para Spark
- `formatOptions`: opciones de formato para el formato especificado. Para obtener información acerca de las opciones de formatos soportados, consulte Opciones de formato de datos.

Ejemplo de origen de streaming de Amazon Kinesis:

```
val data_frame_datasource0 =
glueContext.createDataFrameFromOptions(transformationContext = "datasource0",
 connectionType = "kinesis",
connectionOptions = JsonOptions("""{"streamName": "example_stream", "startingPosition":
 "TRIM_HORIZON", "inferSchema": "true", "classification": "json"}}""""))
```

Ejemplo de origen de streaming de Kafka:

```
val data_frame_datasource0 =
glueContext.createDataFrameFromOptions(transformationContext = "datasource0",
 connectionType = "kafka",
```

```
connectionOptions = JsonOptions("""{"connectionName": "example_connection",
 "topicName": "example_topic", "startingPosition": "earliest", "inferSchema": "false",
 "classification": "json", "schema":"`column1` STRING, `column2` STRING"}"""))
```

forEachBatch

`forEachBatch(frame, batch_function, options)`

Se aplica la `batch_function` transferida a cada microlote que se lee desde el origen de streaming.

- `frame`— El DataFrame que contiene el microlote actual.
- `batch_function`: una función que se aplicará para cada microlote.
- `options`: una recopilación de pares clave-valor que contiene información sobre cómo procesar microlotes. Se requieren las siguientes opciones:
  - `windowSize`: cantidad de tiempo que se debe dedicar al procesamiento de cada lote.
  - `checkpointLocation`: la ubicación donde se almacenan los puntos de verificación para el trabajo de ETL de streaming.
  - `batchMaxRetries`: número máximo de reintentos permitidos para este lote si se genera un error. El valor predeterminado es 3. Esta opción sólo se puede configurar para Glue versión 2.0 y superior.

Ejemplo:

```
glueContext.forEachBatch(data_frame_datasource0, (dataFrame: Dataset[Row], batchId:
 Long) =>
   {
      if (dataFrame.count() > 0)
        {
          val datasource0 = DynamicFrame(glueContext.addIngestionTimeColumns(dataFrame,
 "hour"), glueContext)
          // @type: DataSink
          // @args: [database = "tempdb", table_name = "fromoptionsoutput",
 stream_batch_time = "100 seconds",
          //      stream_checkpoint_location = "s3://from-options-testing-eu-central-1/
fromOptionsOutput/checkpoint/",
          //      transformation_ctx = "datasink1"]
          // @return: datasink1
          // @inputs: [frame = datasource0]
          val options_datasink1 = JsonOptions(
```

```
            Map("partitionKeys" -> Seq("ingest_year", "ingest_month","ingest_day",
 "ingest_hour"),
            "enableUpdateCatalog" -> true))
         val datasink1 = glueContext.getCatalogSink(
            database = "tempdb",
            tableName = "fromoptionsoutput",
            redshiftTmpDir = "",
            transformationContext = "datasink1",
            additionalOptions = options_datasink1).writeDynamicFrame(datasource0)
       }
   }, JsonOptions("""{"windowSize" : "100 seconds",
         "checkpointLocation" : "s3://from-options-testing-eu-central-1/
fromOptionsOutput/checkpoint/"}"""))
```

- definitivamente getCatalogSink

```
def getCatalogSink( database : String,
        tableName : String,
        redshiftTmpDir : String = "",
        transformationContext : String = ""
        additionalOptions: JsonOptions = JsonOptions.empty,
        catalogId: String = null
) : DataSink
```

Crea un DataSink que escribe en una ubicación especificada en una tabla definida en el Data Catalog.

- `database`: nombre de la base de datos en el Data Catalog.
- `tableName`: nombre de la tabla en el Data Catalog.
- `redshiftTmpDir`: directorio provisional que se usará con determinados receptores de datos. Se establece en un valor vacío de forma predeterminada.
- `transformationContext`: contexto de transformación asociado con el receptor que utilizarán los marcadores de trabajo. Se establece en un valor vacío de forma predeterminada.
- `additionalOptions`: opciones adicionales para AWS Glue.
- `catalogId`: ID de catálogo (ID de cuenta) del Data Catalog al que se accede. Cuando el valor es nulo, se utiliza el ID de cuenta predeterminado del intermediario.

Devuelve el `DataSink`.

def getCatalogSource

```
def getCatalogSource( database : String,
                      tableName : String,
                      redshiftTmpDir : String = "",
                      transformationContext : String = ""
                      pushDownPredicate : String = " "
                      additionalOptions: JsonOptions = JsonOptions.empty,
                      catalogId: String = null
                    ) : DataSource
```

Crea un objeto Característica DataSource que lee datos de una definición de tabla en el Data Catalog.

- `database`: nombre de la base de datos en el Data Catalog.
- `tableName`: nombre de la tabla en el Data Catalog.
- `redshiftTmpDir`: directorio provisional que se usará con determinados receptores de datos. Se establece en un valor vacío de forma predeterminada.
- `transformationContext`: contexto de transformación asociado con el receptor que utilizarán los marcadores de trabajo. Se establece en un valor vacío de forma predeterminada.
- `pushDownPredicate`: filtra particiones sin tener que enumerar y leer todos los archivos del conjunto de datos. Para obtener más información, consulte Filtrado previo con predicados de inserción.
- `additionalOptions`: conjunto de pares nombre-valor opcionales. Las opciones posibles incluyen las enumeradas en Tipos de conexión y opciones para ETL en AWS Glue para Spark, excepto por `endpointUrl`, `streamName`, `bootstrap.servers`, `security.protocol`, `topicName`, `classification` y `delimiter`. Otra opción soportada es `catalogPartitionPredicate`:

  `catalogPartitionPredicate`: puede transferir una expresión de catálogo para filtrar en función de las columnas de índice. Esto inserta el filtrado hacia el lado del servidor. Para obtener más información, consulte Índices de partición de AWS Glue. Tenga en cuenta que `push_down_predicate` y `catalogPartitionPredicate` utilizan sintaxis diferentes. El primero utiliza la sintaxis estándar de Spark SQL y el segundo utiliza el analizador JSQL.
- `catalogId`: ID de catálogo (ID de cuenta) del Data Catalog al que se accede. Cuando el valor es nulo, se utiliza el ID de cuenta predeterminado del intermediario.

Devuelve el `DataSource`.

Ejemplo de origen de streaming

```
val data_frame_datasource0 = glueContext.getCatalogSource(
    database = "tempdb",
    tableName = "test-stream-input",
    redshiftTmpDir = "",
    transformationContext = "datasource0",
    additionalOptions = JsonOptions("""{
        "startingPosition": "TRIM_HORIZON", "inferSchema": "false"}""")
    ).getDataFrame()
```

def getJDBCSink

```
def getJDBCSink( catalogConnection : String,
                 options : JsonOptions,
                 redshiftTmpDir : String = "",
                 transformationContext : String = "",
                 catalogId: String = null
               ) : DataSink
```

Crea un DataSink que escribe en una base de datos JDBC especificada en un objeto `Connection` en el Data Catalog. El objeto `Connection` tiene información para conectarse a un receptor de JDBC, incluida la URL, el nombre de usuario, la contraseña, la VPC, la subred y los grupos de seguridad.

- `catalogConnection`: nombre de la conexión en el Data Catalog que contiene la URL de JDBC en la que se va a escribir.
- `options`: cadena de pares de nombre-valor de JSON que proporcionan información adicional que se exige para escribir en un almacén de datos JDBC. Esto incluye:
  - dbtable (obligatorio): nombre de la tabla de JDBC. Para almacenes de datos de JDBC que admiten esquemas dentro de una base de datos, especifique `schema.table-name`. Si no se ha proporcionado un esquema, se usa el esquema "public" predeterminado. El siguiente ejemplo muestra un parámetro de opciones que apunta a un esquema llamado `test` y a una tabla llamada `test_table` en la base de datos `test_db`.

    ```
    options = JsonOptions("""{"dbtable": "test.test_table", "database": "test_db"}""")
    ```

  - database (obligatorio): nombre de la base de datos de JDBC.

  - Todas las opciones adicionales transferidas directamente al escritor SparkSQL de JDBC. Para obtener más información, consulte el artículo acerca del origen de datos Redshift de Spark.
- `redshiftTmpDir`: directorio provisional que se usará con determinados receptores de datos. Se establece en un valor vacío de forma predeterminada.
- `transformationContext`: contexto de transformación asociado con el receptor que utilizarán los marcadores de trabajo. Se establece en un valor vacío de forma predeterminada.
- `catalogId`: ID de catálogo (ID de cuenta) del Data Catalog al que se accede. Cuando el valor es nulo, se utiliza el ID de cuenta predeterminado del intermediario.

Código de ejemplo:

```
getJDBCSink(catalogConnection = "my-connection-name", options =
 JsonOptions("""{"dbtable": "my-jdbc-table", "database": "my-jdbc-db"}"""),
 redshiftTmpDir = "", transformationContext = "datasink4")
```

Devuelve el `DataSink`.

def getSink

```
def getSink( connectionType : String,
             connectionOptions : JsonOptions,
             transformationContext : String = ""
           ) : DataSink
```

Crea una DataSink que escribe datos en un destino como Amazon Simple Storage Service (Amazon S3), JDBC o el catálogo de datos de AWS Glue, o una transmisión de datos de Apache Kafka o Amazon Kinesis.

- `connectionType` — Tipo de la conexión. Consulte the section called “Parámetros de conexión”.
- `connectionOptions`: una cadena de pares de nombre-valor de JSON que proporcionan información adicional para establecer la conexión con el receptor de datos. Consulte the section called “Parámetros de conexión”.
- `transformationContext`: contexto de transformación asociado con el receptor que utilizarán los marcadores de trabajo. Se establece en un valor vacío de forma predeterminada.

Devuelve el `DataSink`.

Formato def getSinkWith

```
def getSinkWithFormat( connectionType : String,
                       options : JsonOptions,
                       transformationContext : String = "",
                       format : String = null,
                       formatOptions : JsonOptions = JsonOptions.empty
                     ) : DataSink
```

Crea una DataSink que escribe datos a destinos tales como Amazon S3, JDBC o el Catálogo de datos o los flujos de datos de Apache Kafka o de Amazon Kinesis. También establece el formato de los datos que se escribirán en el destino.

- `connectionType` — Tipo de la conexión. Consulte the section called “Parámetros de conexión”.
- `options`: cadena de pares de nombre-valor de JSON que proporcionan información adicional para establecer una conexión con el receptor de datos. Consulte the section called “Parámetros de conexión”.
- `transformationContext`: contexto de transformación asociado con el receptor que utilizarán los marcadores de trabajo. Se establece en un valor vacío de forma predeterminada.
- `format`: formato de los datos que se escribirán en el destino.
- `formatOptions`: una cadena de pares de nombre-valor de JSON que proporcionan opciones adicionales para el formato de los datos en el destino. Consulte Opciones de formato de datos.

Devuelve el `DataSink`.

def getSource

```
def getSource( connectionType : String,
               connectionOptions : JsonOptions,
               transformationContext : String = ""
               pushDownPredicate
             ) : DataSource
```

Crea una Característica DataSource que lee datos de una fuente como Amazon S3, JDBC o AWS Glue Data Catalog. También soporta orígenes de datos de streaming de Kafka y Kinesis.

- `connectionType`: tipo de origen de datos. Consulte the section called “Parámetros de conexión”.

- `connectionOptions`: una cadena de pares de nombre-valor de JSON que proporcionan información adicional para establecer una conexión con el origen de datos. Para obtener más información, consulte the section called “Parámetros de conexión”.

  Un origen de streaming de Kinesis requiere las siguientes opciones de conexión: `streamARN`, `startingPosition`, `inferSchema` y `classification`.

  Un origen de streaming de Kafka requiere las siguientes opciones de conexión: `connectionName`, `topicName`, `startingOffsets`, `inferSchema` y `classification`.
- `transformationContext`: contexto de transformación asociado con el receptor que utilizarán los marcadores de trabajo. Se establece en un valor vacío de forma predeterminada.
- `pushDownPredicate`: predicado en columnas de partición.

Devuelve el `DataSource`.

Ejemplo de origen de streaming de Amazon Kinesis:

```
val kinesisOptions = jsonOptions()
data_frame_datasource0 = glueContext.getSource("kinesis",
 kinesisOptions).getDataFrame()

private def jsonOptions(): JsonOptions = {
    new JsonOptions(
      s"""{"streamARN": "arn:aws:kinesis:eu-central-1:123456789012:stream/
fromOptionsStream",
         |"startingPosition": "TRIM_HORIZON",
         |"inferSchema": "true",
         |"classification": "json"}""".stripMargin)
}
```

Ejemplo de origen de streaming de Kafka:

```
val kafkaOptions = jsonOptions()
val data_frame_datasource0 = glueContext.getSource("kafka",
 kafkaOptions).getDataFrame()

private def jsonOptions(): JsonOptions = {
    new JsonOptions(
      s"""{"connectionName": "ConfluentKafka",
         |"topicName": "kafka-auth-topic",
```

```
        |"startingOffsets": "earliest",
        |"inferSchema": "true",
        |"classification": "json"}""".stripMargin)
 }
```

Formato def getSourceWith

```
def getSourceWithFormat( connectionType : String,
                         options : JsonOptions,
                         transformationContext : String = "",
                         format : String = null,
                         formatOptions : JsonOptions = JsonOptions.empty
                       ) : DataSource
```

Crea una Característica DataSource que lee datos de una fuente como Amazon S3, JDBC o AWS Glue Data Catalog, y también establece el formato de los datos almacenados en la fuente.

- `connectionType`: tipo de origen de datos. Consulte the section called “Parámetros de conexión”.
- `options`: una cadena de pares de nombre-valor de JSON que proporciona información adicional para establecer una conexión con el origen de datos. Consulte the section called “Parámetros de conexión”.
- `transformationContext`: contexto de transformación asociado con el receptor que utilizarán los marcadores de flujo de trabajo. Se establece en un valor vacío de forma predeterminada.
- `format`: formato de los datos que se almacenan en el origen. Cuando el `connectionType` es "s3", también puede especificar `format`. Puede ser "avro", "csv", "grokLog", "ion", "json", "xml", "parquet" u "orco".
- `formatOptions`: una cadena de pares de nombre-valor de JSON que proporciona opciones adicionales para analizar los datos en el origen. Consulte Opciones de formato de datos.

Devuelve el `DataSource`.

Ejemplos

Cree un archivo DynamicFrame a partir de una fuente de datos que sea un archivo de valores separados por comas (CSV) en Amazon S3:

```
val datasource0 = glueContext.getSourceWithFormat(
    connectionType="s3",
```

```
    options =JsonOptions(s"""{"paths": [ "s3://csv/nycflights.csv"]}"""),
    transformationContext = "datasource0",
    format = "csv",
    formatOptions=JsonOptions(s"""{"withHeader":"true","separator": ","}""")
    ).getDynamicFrame()
```

Cree una DynamicFrame a partir de una fuente de datos que sea PostgreSQL mediante una conexión JDBC:

```
val datasource0 = glueContext.getSourceWithFormat(
    connectionType="postgresql",
    options =JsonOptions(s"""{
      "url":"jdbc:postgresql://databasePostgres-1.rds.amazonaws.com:5432/testdb",
      "dbtable": "public.company",
      "redshiftTmpDir":"",
      "user":"username",
      "password":"password123"
    }"""),
    transformationContext = "datasource0").getDynamicFrame()
```

Cree DynamicFrame a partir de una fuente de datos que sea MySQL mediante una conexión JDBC:

```
 val datasource0 = glueContext.getSourceWithFormat(
    connectionType="mysql",
    options =JsonOptions(s"""{
      "url":"jdbc:mysql://databaseMysql-1.rds.amazonaws.com:3306/testdb",
      "dbtable": "athenatest_nycflights13_csv",
      "redshiftTmpDir":"",
      "user":"username",
      "password":"password123"
    }"""),
    transformationContext = "datasource0").getDynamicFrame()
```

def getSparkSession

```
def getSparkSession : SparkSession
```

Obtiene el objeto `SparkSession` asociado a esta instancia de GlueContext. Utilice este SparkSession objeto para registrar tablas y UDF para usarlas con las `DataFrame` creadas desde. DynamicFrames

Devuelve el SparkSession.

Definición de startTransaction (iniciar transacción)

```
def startTransaction(readOnly: Boolean):String
```

Inicia una nueva transacción. Llama de forma interna a la API startTransaction de Lake Formation.

- `readOnly`: (booleano) indica si esta transacción debe ser de solo lectura o de lectura y escritura. Se rechazarán las escrituras realizadas con un ID de transacción de solo lectura. No es necesario confirmar las transacciones de solo lectura.

Devuelve el ID de la transacción.

Definición de commitTransaction (confirmar transacción)

```
def commitTransaction(transactionId: String, waitForCommit: Boolean): Boolean
```

Intenta confirmar la transacción especificada. Es posible que se devuelva `commitTransaction` antes de que la transacción haya terminado de confirmarse. Llama de forma interna a la API commitTransaction de Lake Formation.

- `transactionId`: (cadena) la transacción que se confirmará.
- `waitForCommit`: (booleano) determina si se devuelve `commitTransaction` de inmediato. El valor predeterminado es true. Si es falso, `commitTransaction` realiza un sondeo y espera hasta que la transacción se haya confirmado. La cantidad de tiempo de espera se limita a un minuto mediante retroceso exponencial con un máximo de seis reintentos.

Devuelve un valor booleano para indicar si se realizó o no la confirmación.

Definición de cancelTransaction (cancelar transacción)

```
def cancelTransaction(transactionId: String): Unit
```

Intenta cancelar la transacción especificada. Internamente se llama CancelTransactionAPI Lake Formation.

- `transactionId`: (cadena) la transacción que se cancelará.

Devuelve una excepción de `TransactionCommittedException` si la transacción se había confirmado con anterioridad.

def this

```
def this( sc : SparkContext,
          minPartitions : Int,
          targetPartitions : Int )
```

Crea un objeto `GlueContext` con el objeto `SparkContext` especificado, particiones mínimas y particiones de destino.

- `sc` — La `SparkContext`.
- `minPartitions`: número mínimo de particiones.
- `targetPartitions`: número de particiones de destino.

Devuelve el `GlueContext`.

def this

```
def this( sc : SparkContext )
```

Crear un objeto `GlueContext` con el `SparkContext` proporcionado. Establece las particiones mínimas en 10 y las particiones de destino en 20.

- `sc` — La `SparkContext`.

Devuelve el `GlueContext`.

def this

```
def this( sparkContext : JavaSparkContext )
```

Crear un objeto `GlueContext` con el `JavaSparkContext` proporcionado. Establece las particiones mínimas en 10 y las particiones de destino en 20.

- `sparkContext` — La `JavaSparkContext`.

Devuelve el `GlueContext`.

MappingSpec

Paquete: com.amazonaws.services.glue

Case class MappingSpec

```
case class MappingSpec( sourcePath: SchemaPath,
                        sourceType: DataType,
                        targetPath: SchemaPath,
                        targetType: DataTyp
                       ) extends Product4[String, String, String, String] {
  override def _1: String = sourcePath.toString
  override def _2: String = ExtendedTypeName.fromDataType(sourceType)
  override def _3: String = targetPath.toString
  override def _4: String = ExtendedTypeName.fromDataType(targetType)
}
```

- `sourcePath`: `SchemaPath` del campo de origen.
- `sourceType`: `DataType` del campo de origen.
- `targetPath`: `SchemaPath` del campo de destino.
- `targetType`: `DataType` del campo de destino.

Un objeto `MappingSpec` especifica un mapeo desde una ruta y un tipo de datos de origen a una ruta y un tipo de datos de destino. El valor en la ruta de origen en el marco de origen aparece en el marco de destino en la ruta de destino. El tipo de datos de origen se convierte al de destino.

Se extiende desde `Product4` para que usted pueda gestionar todos los `Product4` de su interfaz `applyMapping`.

Objeto MappingSpec

```
object MappingSpec
```

El objeto `MappingSpec` tiene los siguientes miembros:

Val orderingByTarget

```
val orderingByTarget: Ordering[MappingSpec]
```

Def apply

```
def apply( sourcePath : String,
           sourceType : DataType,
           targetPath : String,
           targetType : DataType
         ) : MappingSpec
```

Crea un `MappingSpec`.

- `sourcePath`: representación de cadena de la ruta de origen.
- `sourceType`: `DataType` de origen.
- `targetPath`: representación de cadena de la ruta de destino.
- `targetType`: el objeto `DataType` de destino.

Devuelve `MappingSpec`.

Def apply

```
def apply( sourcePath : String,
           sourceTypeString : String,
           targetPath : String,
           targetTypeString : String
         ) : MappingSpec
```

Crea un `MappingSpec`.

- `sourcePath`: representación de cadena de la ruta de origen.
- `sourceType`: representación de cadena del tipo de datos de origen.
- `targetPath`: representación de cadena de la ruta de destino.
- `targetType`: representación de cadena del tipo de datos de destino.

Devuelve un objeto MappingSpec.

Def apply

```
def apply( product : Product4[String, String, String, String] ) : MappingSpec
```

Crea un `MappingSpec`.

- `product`: el elemento `Product4` de la ruta de origen, el tipo de datos de origen, la ruta de destino y el tipo de datos de destino.

Devuelve `MappingSpec`.

API ResolveSpec Scala de AWS Glue

Temas

- Objeto ResolveSpec
- Case class ResolveSpec

Paquete: com.amazonaws.services.glue

Objeto ResolveSpec

ResolveSpec

```
object ResolveSpec
```

Def

```
def apply( path : String,
           action : String
         ) : ResolveSpec
```

Crea un `ResolveSpec`.

- `path`: representación de cadena del campo de elección que se debe resolver.
- `action`: acción de resolución. La acción puede ser una de las siguientes: `Project`, `KeepAsStruct` o `Cast`.

Devuelve el `ResolveSpec`.

Def

```
def apply( product : Product2[String, String] ) : ResolveSpec
```

Crea un `ResolveSpec`.

- `product`: `Product2` de: ruta de origen, acción de resolución.

Devuelve el `ResolveSpec`.

Case class ResolveSpec

```
case class ResolveSpec extends Product2[String, String]  (
           path : SchemaPath,
           action : String )
```

Crea un `ResolveSpec`.

- `path`: `SchemaPath` del campo de elección que se debe resolver.
- `action`: acción de resolución. La acción puede ser una de las siguientes: `Project`, `KeepAsStruct` o `Cast`.

Métodos def de ResolveSpec

```
def _1 : String
```

```
def _2 : String
```

API ArrayNode Scala de AWS Glue

Paquete: com.amazonaws.services.glue.types

Case class ArrayNode

ArrayNode

```
case class ArrayNode extends DynamicNode  (
           value : ArrayBuffer[DynamicNode] )
```

Métodos def de ArrayNode

```
def add( node : DynamicNode )
```

```
def clone
```

```
def equals( other : Any )
```

```
def get( index : Int ) : Option[DynamicNode]
```

```
def getValue
```

```
def hashCode : Int
```

```
def isEmpty : Boolean
```

```
def nodeType
```

```
def remove( index : Int )
```

```
def this
```

```
def toIterator : Iterator[DynamicNode]
```

```
def toJson : String
```

```
def update( index : Int,
            node : DynamicNode )
```

API BinaryNode Scala de AWS Glue

Paquete: com.amazonaws.services.glue.types

Case class BinaryNode

BinaryNode

```
case class BinaryNode extends ScalarNode(value, TypeCode.BINARY)  (
           value : Array[Byte] )
```

Campos val de BinaryNode

- `ordering`

Métodos def de BinaryNode

```
def clone
```

```
def equals( other : Any )
```

```
def hashCode : Int
```

API BooleanNode Scala de AWS Glue

Paquete: com.amazonaws.services.glue.types

Case class BooleanNode

BooleanNode

```
case class BooleanNode extends ScalarNode(value, TypeCode.BOOLEAN)  (
           value : Boolean )
```

Campos val de BooleanNode

- `ordering`

Métodos def de BooleanNode

```
def equals( other : Any )
```

API ByteNode Scala de AWS Glue

Paquete: com.amazonaws.services.glue.types

Case class ByteNode

ByteNode

```
case class ByteNode extends ScalarNode(value, TypeCode.BYTE)  (
```

```
            value : Byte )
```

Campos val de ByteNode

- `ordering`

Métodos def de ByteNode

```
def equals( other : Any )
```

API DateNode Scala de AWS Glue

Paquete: com.amazonaws.services.glue.types

Case class DateNode

DateNode

```
case class DateNode extends ScalarNode(value, TypeCode.DATE)  (
            value : Date )
```

Campos val de DateNode

- `ordering`

Métodos def de DateNode

```
def equals( other : Any )
```

```
def this( value : Int )
```

API DecimalNode Scala de AWS Glue

Paquete: com.amazonaws.services.glue.types

Case class DecimalNode

DecimalNode

```
case class DecimalNode extends ScalarNode(value, TypeCode.DECIMAL)  (
```

```
            value : BigDecimal )
```

Campos val de DecimalNode

- `ordering`

Métodos def de DecimalNode

```
def equals( other : Any )
```

```
def this( value : Decimal )
```

API DoubleNode Scala de AWS Glue

Paquete: com.amazonaws.services.glue.types

Case class DoubleNode

DoubleNode

```
case class DoubleNode extends ScalarNode(value, TypeCode.DOUBLE)  (
            value : Double )
```

Campos val de DoubleNode

- `ordering`

Métodos def de DoubleNode

```
def equals( other : Any )
```

API DynamicNode Scala de AWS Glue

Temas

- Clase DynamicNode
- Objeto DynamicNode

Paquete: com.amazonaws.services.glue.types

Clase DynamicNode

DynamicNode

```
class DynamicNode extends Serializable with Cloneable
```

Métodos def de DynamicNode

```
def getValue : Any
```

Obtenga el valor sin formato y vincúlelo al registro actual:

```
def nodeType : TypeCode
```

```
def toJson : String
```

Método de depuración:

```
def toRow( schema : Schema,
           options : Map[String, ResolveOption]
         ) : Row
```

```
def typeName : String
```

Objeto DynamicNode

DynamicNode

```
object DynamicNode
```

Métodos def de DynamicNode

```
def quote( field : String,
           useQuotes : Boolean
         ) : String
```

```
def quote( node : DynamicNode,
           useQuotes : Boolean
         ) : String
```

Clase de EvaluateDataQuality

> Calidad de datos de AWS Glue se encuentra en versión preliminar para AWS Glue, por lo que es una característica que está sujeta a cambios.

Paquete: com.amazonaws.services.glue.dq

```
object EvaluateDataQuality
```

Def apply

```
def apply(frame: DynamicFrame,
            ruleset: String,
            publishingOptions: JsonOptions = JsonOptions.empty): DynamicFrame
```

Evalúa un conjunto de reglas de calidad de datos en relación con un `DynamicFrame` y devuelve un `DynamicFrame` nuevo con los resultados de la evaluación. Para obtener más información sobre Calidad de datos de AWS Glue, consulte AWS Glue Calidad de los datos.

- `frame`: el objeto `DynamicFrame` del que quiere evaluar la calidad de los datos.
- `ruleset`: un conjunto de reglas de lenguaje de definición de calidad de datos (DQDL) en formato de cadena. Para obtener más información sobre DQDL, consulte la guía de Referencia del lenguaje de definición de calidad de datos (DQDL).
- `publishingOptions`: un diccionario que especifica las siguientes opciones para publicar los resultados y las métricas de la evaluación:
  - `dataQualityEvaluationContext`: una cadena que especifica el espacio de nombres en el que AWS Glue debe publicar las métricas de Amazon CloudWatch y los resultados de calidad de los datos. Las métricas agregadas aparecen en CloudWatch, mientras que los resultados completos aparecen en la interfaz de AWS Glue Studio.
    - Requerido: no
    - Valor predeterminado: `default_context`
  - `enableDataQualityCloudWatchMetrics`: especifica si los resultados de la evaluación de la calidad de los datos deben publicarse en CloudWatch. Especifique un espacio de nombres para las métricas mediante la opción `dataQualityEvaluationContext`.
    - Requerido: no

  - Valor predeterminado: False
- `enableDataQualityResultsPublishing`: especifica si los resultados de calidad de los datos deben estar visibles en la pestaña Data Quality (Calidad de datos) de la interfaz de AWS Glue Studio.
  - Requerido: no
  - Valor predeterminado: true
- `resultsS3Prefix`: especifica la ubicación de Amazon S3 en la que AWS Glue puede escribir los resultados de la evaluación de la calidad de los datos.
  - Requerido: no
  - Valor predeterminado: "" (cadena vacía)

Ejemplo

El siguiente código de ejemplo muestra cómo evaluar la calidad de los datos de un `DynamicFrame` antes de hacer una transformación de `SelectFields`. El script verifica que se cumplan todas las reglas de calidad de datos antes de intentar la transformación.

```
import com.amazonaws.services.glue.GlueContext
import com.amazonaws.services.glue.MappingSpec
import com.amazonaws.services.glue.errors.CallSite
import com.amazonaws.services.glue.util.GlueArgParser
import com.amazonaws.services.glue.util.Job
import com.amazonaws.services.glue.util.JsonOptions
import org.apache.spark.SparkContext
import scala.collection.JavaConverters._
import com.amazonaws.services.glue.dq.EvaluateDataQuality

object GlueApp {
  def main(sysArgs: Array[String]) {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)
    // @params: [JOB_NAME]
    val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB_NAME").toArray)
    Job.init(args("JOB_NAME"), glueContext, args.asJava)

    // Create DynamicFrame with data
    val Legislators_Area = glueContext.getCatalogSource(database="legislators",
 tableName="areas_json", transformationContext="S3bucket_node1").getDynamicFrame()
```

```
    // Define data quality ruleset
    val DQ_Ruleset = """
      Rules = [ColumnExists "id"]
    """

    // Evaluate data quality
    val DQ_Results = EvaluateDataQuality.apply(frame=Legislators_Area,
 ruleset=DQ_Ruleset, publishingOptions=JsonOptions("""{"dataQualityEvaluationContext":
 "Legislators_Area", "enableDataQualityMetrics": "true",
 "enableDataQualityResultsPublishing": "true"}"""))
    assert(DQ_Results.filter(_.getField("Outcome").contains("Failed")).count == 0,
 "Failing DQ rules for Legislators_Area caused the job to fail.")

    // Script generated for node Select Fields
    val SelectFields_Results = Legislators_Area.selectFields(paths=Seq("id", "name"),
 transformationContext="Legislators_Area")

    Job.commit()
  }
}
```

API FloatNode Scala de AWS Glue

Paquete: com.amazonaws.services.glue.types

Case class FloatNode

FloatNode

```
case class FloatNode extends ScalarNode(value, TypeCode.FLOAT)  (
           value : Float )
```

Campos val de FloatNode

- `ordering`

Métodos def de FloatNode

```
def equals( other : Any )
```

Clase FillMissingValues

Paquete: com.amazonaws.services.glue.ml

```
object FillMissingValues
```

Def apply

```
def apply(frame: DynamicFrame,
          missingValuesColumn: String,
          outputColumn: String = "",
          transformationContext: String = "",
          callSite: CallSite = CallSite("Not provided", ""),
          stageThreshold: Long = 0,
          totalThreshold: Long = 0): DynamicFrame
```

Rellena los valores que faltan de un marco dinámico en una columna especificada y devuelve un nuevo marco con estimaciones en una nueva columna. Para filas sin valores faltantes, el valor de la columna especificada se duplica en la nueva columna.

- `frame`: DynamicFrame en el que rellenar los valores que faltan. Obligatorio.
- `missingValuesColumn`: la columna que contiene valores faltantes (valores `null` y cadenas vacías). Obligatorio.
- `outputColumn`: el nombre de la nueva columna que contendrá valores estimados para todas las filas cuyo valor faltaba. Opcional; el valor predeterminado es `missingValuesColumn` con el sufijo `"_filled"`.
- `transformationContext`: una cadena única que se utiliza para identificar la información de estado (opcional).
- `callSite`: se utiliza para proporcionar información de contexto para los informes de error. (Opcional).
- `stageThreshold`: el número máximo de errores que se pueden producir en la transformación antes de que se determine que es errónea (opcional, el valor predeterminado es cero).
- `totalThreshold`: el número máximo de errores que se pueden producir en total antes de que se determine que el proceso es erróneo (opcional, el valor predeterminado es cero).

Devuelve un nuevo marco dinámico con una columna adicional que contiene estimaciones para filas con valores faltantes y el valor actual para otras filas.

Clase FindMatches

Paquete: com.amazonaws.services.glue.ml

```
object FindMatches
```

Def apply

```
def apply(frame: DynamicFrame,
            transformId: String,
            transformationContext: String = "",
            callSite: CallSite = CallSite("Not provided", ""),
            stageThreshold: Long = 0,
            totalThreshold: Long = 0,
            enforcedMatches: DynamicFrame = null): DynamicFrame,
   computeMatchConfidenceScores: Boolean
```

Busque coincidencias en un marco de entrada y devuelva un nuevo marco con una nueva columna que contenga un ID único por grupo de coincidencias.

- `frame`: DynamicFrame en el que buscar coincidencias. Obligatorio.
- `transformId`: ID único asociado a la transformación FindMatches que se aplicará en el marco de entrada. Obligatorio.
- `transformationContext`: identificador de este `DynamicFrame`. Se utiliza `transformationContext` como clave para el estado de marcador de trabajo que se conserva de una ejecución a otra. Opcional.
- `callSite`: se utiliza para proporcionar información de contexto para los informes de error. Estos valores se establecen automáticamente cuando se realiza la llamada desde Python. Opcional.
- `stageThreshold`: número máximo de registros de errores permitidos en el cálculo de este `DynamicFrame` antes de generar una excepción, sin incluir los registros presentes en el `DynamicFrame` anterior. Opcional. El rol predeterminado es cero.
- `totalThreshold`: número máximo registros de errores totales antes de que se genere una excepción, incluidos los de marcos anteriores. Opcional. El rol predeterminado es cero.
- `enforcedMatches`: el marco para las coincidencias forzadas. Opcional. El valor predeterminado es `null`.
- `computeMatchConfidenceScores`: valor booleano que indica si se calcula una puntuación de confianza para cada grupo de registros coincidentes. Opcional. El valor predeterminado es false.

Devuelve un nuevo marco dinámico con un identificador único asignado a cada grupo de registros coincidentes.

Clase FindIncrementalMatches

Paquete: com.amazonaws.services.glue.ml

```
object FindIncrementalMatches
```

Def apply

```
apply(existingFrame: DynamicFrame,
            incrementalFrame: DynamicFrame,
            transformId: String,
            transformationContext: String = "",
            callSite: CallSite = CallSite("Not provided", ""),
            stageThreshold: Long = 0,
            totalThreshold: Long = 0,
            enforcedMatches: DynamicFrame = null): DynamicFrame,
   computeMatchConfidenceScores: Boolean
```

Busca coincidencias entre los marcos existentes y progresivos y devuelve un nuevo marco con una columna que contiene un ID único por grupo de coincidencias.

- `existingframe`: un marco existente al que se le ha asignado un ID coincidente para cada grupo. Obligatorio.
- `incrementalframe`: un marco progresivo utilizado para encontrar coincidencias con el marco existente. Obligatorio.
- `transformId`: un ID único asociado a la transformación FindIncrementalMatches que se aplicará a los marcos de entrada. Obligatorio.
- `transformationContext`: identificador de este `DynamicFrame`. Se utiliza `transformationContext` como clave para el estado de marcador de trabajo que se conserva de una ejecución a otra. Opcional.
- `callSite`: se utiliza para proporcionar información de contexto para los informes de error. Estos valores se establecen automáticamente cuando se realiza la llamada desde Python. Opcional.
- `stageThreshold`: número máximo de registros de errores permitidos en el cálculo de este `DynamicFrame` antes de generar una excepción, sin incluir los registros presentes en el `DynamicFrame` anterior. Opcional. El rol predeterminado es cero.

- `totalThreshold`: número máximo registros de errores totales antes de que se genere una excepción, incluidos los de marcos anteriores. Opcional. El rol predeterminado es cero.
- `enforcedMatches`: el marco para las coincidencias forzadas. Opcional. El valor predeterminado es `null`.
- `computeMatchConfidenceScores`: valor booleano que indica si se calcula una puntuación de confianza para cada grupo de registros coincidentes. Opcional. El valor predeterminado es false.

Devuelve un nuevo marco dinámico con un identificador único asignado a cada grupo de registros coincidentes.

API IntegerNode Scala de AWS Glue

Paquete: com.amazonaws.services.glue.types

Case class IntegerNode

IntegerNode

```
case class IntegerNode extends ScalarNode(value, TypeCode.INT)  (
           value : Int )
```

Campos val de IntegerNode

- `ordering`

Métodos def de IntegerNode

```
def equals( other : Any )
```

API LongNode Scala de AWS Glue

Paquete: com.amazonaws.services.glue.types

Case class LongNode

LongNode

```
case class LongNode extends ScalarNode(value, TypeCode.LONG)  (
           value : Long )
```

Campos val de LongNode

- `ordering`

Métodos def de LongNode

```
def equals( other : Any )
```

API MapLikeNode Scala de AWS Glue

Paquete: com.amazonaws.services.glue.types

Case class MapLikeNode

MapLikeNode

```
class MapLikeNode extends DynamicNode  (
           value : mutable.Map[String, DynamicNode] )
```

Métodos def de MapLikeNode

```
def clear : Unit
```

```
def get( name : String ) : Option[DynamicNode]
```

```
def getValue
```

```
def has( name : String ) : Boolean
```

```
def isEmpty : Boolean
```

```
def put( name : String,
         node : DynamicNode
       ) : Option[DynamicNode]
```

```
def remove( name : String ) : Option[DynamicNode]
```

```
def toIterator : Iterator[(String, DynamicNode)]
```

```
def toJson : String
```

```
def toJson( useQuotes : Boolean ) : String
```

Ejemplo: Teniendo en cuenta este JSON:

```
{"foo": "bar"}
```

Si `useQuotes == true`, `toJson` devuelve `{"foo": "bar"}`. Si `useQuotes == false`, `toJson` devuelve `{foo: bar}` @return.

API MapNode Scala de AWS Glue

Paquete: com.amazonaws.services.glue.types

Case class MapNode

MapNode

```
case class MapNode extends MapLikeNode(value)  (
           value : mutable.Map[String, DynamicNode] )
```

Métodos def de MapNode

```
def clone
```

```
def equals( other : Any )
```

```
def hashCode : Int
```

```
def nodeType
```

```
def this
```

API NullNode Scala de AWS Glue

Temas

- Clase NullNode
- Case object NullNode

Paquete: com.amazonaws.services.glue.types

Clase NullNode

NullNode

```
class NullNode
```

Case object NullNode

NullNode

```
case object NullNode extends NullNode
```

API ObjectNode Scala de AWS Glue

Temas

- Objeto ObjectNode
- Case class ObjectNode

Paquete: com.amazonaws.services.glue.types

Objeto ObjectNode

ObjectNode

```
object ObjectNode
```

Métodos def de ObjectNode

```
def apply( frameKeys : Set[String],
           v1 : mutable.Map[String, DynamicNode],
           v2 : mutable.Map[String, DynamicNode],
```

```
            resolveWith : String
          ) : ObjectNode
```

Case class ObjectNode

ObjectNode

```
case class ObjectNode extends MapLikeNode(value)  (
            val value : mutable.Map[String, DynamicNode] )
```

Métodos def de ObjectNode

```
def clone
```

```
def equals( other : Any )
```

```
def hashCode : Int
```

```
def nodeType
```

```
def this
```

API ScalarNode Scala de AWS Glue

Temas

- Clase ScalarNode
- Objeto ScalarNode

Paquete: com.amazonaws.services.glue.types

Clase ScalarNode

ScalarNode

```
class ScalarNode extends DynamicNode  (
            value : Any,
            scalarType : TypeCode )
```

Métodos def de ScalarNode

```
def compare( other : Any,
             operator : String
           ) : Boolean
```

```
def getValue
```

```
def hashCode : Int
```

```
def nodeType
```

```
def toJson
```

Objeto ScalarNode

ScalarNode

```
object ScalarNode
```

Métodos def de ScalarNode

```
def apply( v : Any ) : DynamicNode
```

```
def compare( tv : Ordered[T],
             other : T,
             operator : String
           ) : Boolean
```

```
def compareAny( v : Any,
                y : Any,
                o : String )
```

```
def withEscapedSpecialCharacters( jsonToEscape : String ) : String
```

API ShortNode Scala de AWS Glue

Paquete: com.amazonaws.services.glue.types

Case class ShortNode

ShortNode

```
case class ShortNode extends ScalarNode(value, TypeCode.SHORT)  (
           value : Short )
```

Campos val de ShortNode

- `ordering`

Métodos def de ShortNode

```
def equals( other : Any )
```

API StringNode Scala de AWS Glue

Paquete: com.amazonaws.services.glue.types

Case class StringNode

StringNode

```
case class StringNode extends ScalarNode(value, TypeCode.STRING)  (
           value : String )
```

Campos val de StringNode

- `ordering`

Métodos def de StringNode

```
def equals( other : Any )
```

```
def this( value : UTF8String )
```

API TimestampNode Scala de AWS Glue

Paquete: com.amazonaws.services.glue.types

Case class TimestampNode

TimestampNode

```
case class TimestampNode extends ScalarNode(value, TypeCode.TIMESTAMP)  (
           value : Timestamp )
```

Campos val de TimestampNode

- `ordering`

Métodos def de TimestampNode

```
def equals( other : Any )
```

```
def this( value : Long )
```

API GlueArgParser Scala de AWS Glue

Paquete: com.amazonaws.services.glue.util

Objeto GlueArgParser

GlueArgParser

```
object GlueArgParser
```

Esto es estrictamente coherente con la versión para Python de `utils.getResolvedOptions` del paquete `AWSGlueDataplanePython`.

Métodos def de GlueArgParser

```
def getResolvedOptions( args : Array[String],
                        options : Array[String]
                      ) : Map[String, String]
```

```
def initParser( userOptionsSet : mutable.Set[String] ) : ArgumentParser
```

Example Recuperación de los argumentos que se pasan a un trabajo

Para recuperar los argumentos del trabajo, puede usar el método `getResolvedOptions`. Considere el siguiente ejemplo, que recupera un argumento de trabajo denominado `aws_region`.

```
val args = GlueArgParser.getResolvedOptions(sysArgs,
 Seq("JOB_NAME","aws_region").toArray)
Job.init(args("JOB_NAME"), glueContext, args.asJava)
val region = args("aws_region")
println(region)
```

API Job Scala de AWS Glue

Paquete: com.amazonaws.services.glue.util

Objeto de trabajo

Trabajo

```
object Job
```

Métodos def de trabajo

```
def commit
```

```
def init( jobName : String,
          glueContext : GlueContext,
          args : java.util.Map[String, String] = Map[String, String]().asJava
        ) : this.type
```

```
def init( jobName : String,
          glueContext : GlueContext,
          endpoint : String,
          args : java.util.Map[String, String]
        ) : this.type
```

```
def isInitialized
```

```
def reset
```

```
def runId
```

Características y optimizaciones para programar AWS Glue para los scripts de Spark ETL

En las siguientes secciones se describen las técnicas y los valores que se aplican de manera general a AWS Glue para la programación de Spark ETL (extracción, transformación y carga).

Temas

- Tipos de conexión y opciones para ETL en AWS Glue para Spark
- Opciones de formato de datos para las entradas y las salidas en AWS Glue para Spark
- Soporte de AWS Glue Data Catalog para trabajos de Spark SQL
- Uso de marcadores de trabajo
- Uso de la detección de datos confidenciales fuera de AWS Glue Studio
- API de Visual Job de AWS Glue

Tipos de conexión y opciones para ETL en AWS Glue para Spark

En AWS Glue para Spark, varios métodos y transformaciones de PySpark y Scala especifican el tipo de conexión mediante un parámetro `connectionType`. Especifican las opciones de conexión mediante un parámetro `connectionOptions` o `options`.

El parámetro `connectionType` puede adoptar los valores que se muestran en la tabla siguiente. Los valores del parámetro `connectionOptions` (u `options`) asociados para cada tipo se documentan en las secciones siguientes. Salvo que se indique lo contrario, los parámetros se aplican cuando la conexión se utiliza como origen o receptor.

Para obtener un código de muestra que ilustra la configuración y que utiliza las opciones de conexión, consulte la página de inicio de cada tipo de conexión.

`connectionType`	Se conecta a
dynamodb	Base de datos Amazon DynamoDB
kinesis	Amazon Kinesis Data Streams

`connectionType`	Se conecta a
s3	Amazon S3
documentdb	Base de datos de Amazon DocumentDB (con compatibilidad con MongoDB)
opensearch	Amazon OpenSearch Service.
redshift	Base de datos de Amazon Redshift
kafka	Kafka o Amazon Managed Streaming for Apache Kafka
azurecosmos	Azure Cosmos para NoSQL.
azuresql	Azure SQL.
bigquery	Google BigQuery.
mongodb	Base de datos MongoDB, incluido MongoDB Atlas.
sqlserver	Base de datos de Microsoft SQL Server (consulte Conexiones de JDBC)
mysql	Base de datos de MySQL (consulte Conexiones de JDBC)
oracle	Base de datos de Oracle (consulte Conexiones de JDBC)
postgresql	Base de datos de PostgreSQL (consulte Conexiones de JDBC)
saphana	SAP HANA.
Snowflake	Lago de datos Snowflake
teradata	Teradata Vantage.
vertica	Vertica.
custom.*	Almacenes de datos Spark, Athena o JDBC (consulte Valores de conexiones de tipo personalizada y AWS Marketplace

`connectionType`	Se conecta a
marketplace.*	Almacenes de datos Spark, Athena o JDBC (consulte Valores de conexiones de tipo personalizada y AWS Marketplace)

Conexiones de DynamoDB

Puede utilizar AWS Glue para Spark para leer y escribir en tablas de DynamoDB en AWS Glue. Para conectarse a DynamoDB, utilice los permisos de IAM adjuntos a su trabajo de AWSGlue. AWS Glue admite la escritura de datos en otra cuenta de tablas de DynamoDB de AWS. Para obtener más información, consulte the section called “Acceso entre cuentas y entre regiones a tablas de DynamoDB”.

Además del conector de ETL de DynamoDB de AWS Glue, puede leer en DynamoDB por medio del conector de exportación de DynamoDB, que invoca una solicitud `ExportTableToPointInTime` de DynamoDB y la almacena en una ubicación de Amazon S3 que se proporcione, con el formato de DynamoDB JSON. Después, AWS Glue crea un objeto DynamicFrame mediante la lectura de los datos desde la ubicación de exportación de Amazon S3.

El escritor de DynamoDB está disponible en la versión AWS Glue 1.0 o versiones posteriores. El conector de exportación de DynamoDB de AWS Glue está disponible en la versión 2.0 o versiones posteriores de AWS Glue.

Para obtener más información sobre DynamoDB, consulte la documentación de Amazon DynamoDB.

> **Note**
>
> El lector de ETL de DynamoDB no es compatible con filtros ni predicados de inserción.

Configuración de las conexiones DynamoDB

Para conectarse a DynamoDB desde AWS Glue, conceda permiso al rol de IAM asociado a su trabajo de AWS Glue para interactuar con DynamoDB. Para obtener más información sobre los permisos necesarios para leer o escribir en DynamoDB, consulte Acciones, recursos y claves de condición para DynamoDB en la documentación de IAM.

En las siguientes situaciones, es posible que necesite una configuración adicional:

- Cuando utilice el conector de exportación de DynamoDB, tendrá que configurar IAM para que su trabajo pueda solicitar exportaciones de tablas de DynamoDB. Además, tendrá que identificar un bucket de Amazon S3 para la exportación y proporcionar los permisos adecuados en IAM para que DynamoDB escriba en él y para que su trabajo de AWS Glue pueda leerlo. Para obtener más información, consulte Solicitar una exportación de tabla en DynamoDB.
- Si su trabajo de AWS Glue tiene requisitos de conectividad de Amazon VPC específicos, utilice el tipo de conexión `NETWORK` de AWS Glue para proporcionar opciones de red. Como el acceso a DynamoDB está autorizado por IAM, no es necesario utilizar un tipo de conexión de DynamoDB de AWS Glue.

Lectura y escritura en DynamoDB

En los siguientes ejemplos de código, se muestra cómo leer (con el conector de ETL) de tablas de DynamoDB y escribir en ellas. Demuestran la lectura de una tabla y la escritura en otra tabla.

Python

```
import sys
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job
from awsglue.utils import getResolvedOptions

args = getResolvedOptions(sys.argv, ["JOB_NAME"])
glue_context= GlueContext(SparkContext.getOrCreate())
job = Job(glue_context)
job.init(args["JOB_NAME"], args)

dyf = glue_context.create_dynamic_frame.from_options(
    connection_type="dynamodb",
    connection_options={"dynamodb.input.tableName": test_source,
        "dynamodb.throughput.read.percent": "1.0",
        "dynamodb.splits": "100"
    }
)
print(dyf.getNumPartitions())

glue_context.write_dynamic_frame_from_options(
    frame=dyf,
    connection_type="dynamodb",
    connection_options={"dynamodb.output.tableName": test_sink,
```

```
        "dynamodb.throughput.write.percent": "1.0"
    }
)

job.commit()
```

Scala

```
import com.amazonaws.services.glue.GlueContext
import com.amazonaws.services.glue.util.GlueArgParser
import com.amazonaws.services.glue.util.Job
import com.amazonaws.services.glue.util.JsonOptions
import com.amazonaws.services.glue.DynamoDbDataSink
import org.apache.spark.SparkContext
import scala.collection.JavaConverters._


object GlueApp {

  def main(sysArgs: Array[String]): Unit = {
    val glueContext = new GlueContext(SparkContext.getOrCreate())
    val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB_NAME").toArray)
    Job.init(args("JOB_NAME"), glueContext, args.asJava)

    val dynamicFrame = glueContext.getSourceWithFormat(
      connectionType = "dynamodb",
      options = JsonOptions(Map(
        "dynamodb.input.tableName" -> test_source,
        "dynamodb.throughput.read.percent" -> "1.0",
        "dynamodb.splits" -> "100"
      ))
    ).getDynamicFrame()

    print(dynamicFrame.getNumPartitions())

    val dynamoDbSink: DynamoDbDataSink =  glueContext.getSinkWithFormat(
      connectionType = "dynamodb",
      options = JsonOptions(Map(
        "dynamodb.output.tableName" -> test_sink,
        "dynamodb.throughput.write.percent" -> "1.0"
      ))
    ).asInstanceOf[DynamoDbDataSink]
```

```
    dynamoDbSink.writeDynamicFrame(dynamicFrame)

    Job.commit()
  }

}
```

Uso del conector de exportación de DynamoDB

El conector de exportación funciona mejor que el conector de ETL cuando el tamaño de la tabla de DynamoDB es superior a 80 GB. Además, dado que la solicitud de exportación se lleva a cabo fuera de los procesos de Spark en un trabajo de AWS Glue, se puede habilitar el escalado automático de los trabajos de AWS Glue para guardar el uso de DPU durante la solicitud de exportación. Con el conector de exportación, tampoco es necesario configurar el número de divisiones del paralelismo del ejecutor de Spark o el porcentaje de lectura de rendimiento de DynamoDB.

> **Note**
>
> DynamoDB tiene requisitos específicos para invocar las solicitudes `ExportTableToPointInTime`. Para obtener más información, consulte Solicitud de una exportación de tabla en DynamoDB. Por ejemplo, debe estar habilitada la restauración a un momento dado (PITR) en la tabla para utilizar este conector. El conector de DynamoDB también es compatible con el cifrado de AWS KMS para las exportaciones de DynamoDB a Amazon S3. Proporcionar la configuración de seguridad en la configuración del trabajo de AWS Glue habilita el cifrado de AWS KMS para una exportación de DynamoDB. La clave de KMS debe estar en la misma región que el bucket de Simple Storage Service (Amazon S3). Tenga en cuenta que se aplican cargos adicionales por la exportación de DynamoDB y los costos de almacenamiento de Simple Storage Service (Amazon S3). Los datos exportados en Simple Storage Service (Amazon S3) se mantienen cuando finaliza la ejecución de un trabajo para que pueda reutilizarlos sin exportaciones adicionales de DynamoDB. Un requisito para utilizar este conector es que la recuperación en un momento dado (PITR) esté habilitada para la tabla.
> El conector de ETL y el conector de exportación de DynamoDB no son compatibles con la aplicación de filtros o predicados de inserción en el origen de DynamoDB.

Los siguientes ejemplos de código muestran cómo leer desde particiones (a través del conector de exportación) e imprimir el número de estas.

Python

```
import sys
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job
from awsglue.utils import getResolvedOptions

args = getResolvedOptions(sys.argv, ["JOB_NAME"])
glue_context= GlueContext(SparkContext.getOrCreate())
job = Job(glue_context)
job.init(args["JOB_NAME"], args)

dyf = glue_context.create_dynamic_frame.from_options(
    connection_type="dynamodb",
    connection_options={
        "dynamodb.export": "ddb",
        "dynamodb.tableArn": test_source,
        "dynamodb.s3.bucket": bucket_name,
        "dynamodb.s3.prefix": bucket_prefix,
        "dynamodb.s3.bucketOwner": account_id_of_bucket,
    }
)
print(dyf.getNumPartitions())

job.commit()
```

Scala

```
import com.amazonaws.services.glue.GlueContext
import com.amazonaws.services.glue.util.GlueArgParser
import com.amazonaws.services.glue.util.Job
import com.amazonaws.services.glue.util.JsonOptions
import com.amazonaws.services.glue.DynamoDbDataSink
import org.apache.spark.SparkContext
import scala.collection.JavaConverters._


object GlueApp {

  def main(sysArgs: Array[String]): Unit = {
    val glueContext = new GlueContext(SparkContext.getOrCreate())
    val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB_NAME").toArray)
```

```
    Job.init(args("JOB_NAME"), glueContext, args.asJava)

    val dynamicFrame = glueContext.getSourceWithFormat(
      connectionType = "dynamodb",
      options = JsonOptions(Map(
        "dynamodb.export" -> "ddb",
        "dynamodb.tableArn" -> test_source,
        "dynamodb.s3.bucket" -> bucket_name,
        "dynamodb.s3.prefix" -> bucket_prefix,
        "dynamodb.s3.bucketOwner" -> account_id_of_bucket,
      ))
    ).getDynamicFrame()

    print(dynamicFrame.getNumPartitions())

    Job.commit()
  }

}
```

Estos ejemplos muestran cómo realizar la lectura desde particiones (a través del conector de exportación) e imprimir el número de estas desde una tabla de Data Catalog de AWS Glue que tenga una clasificación dynamodb:

Python

```
import sys
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job
from awsglue.utils import getResolvedOptions

args = getResolvedOptions(sys.argv, ["JOB_NAME"])
glue_context= GlueContext(SparkContext.getOrCreate())
job = Job(glue_context)
job.init(args["JOB_NAME"], args)

dynamicFrame = glue_context.create_dynamic_frame.from_catalog(
        database=catalog_database,
        table_name=catalog_table_name,
        additional_options={
            "dynamodb.export": "ddb",
```

```
            "dynamodb.s3.bucket": s3_bucket,
            "dynamodb.s3.prefix": s3_bucket_prefix
        }
    )
print(dynamicFrame.getNumPartitions())

job.commit()
```

Scala

```
import com.amazonaws.services.glue.GlueContext
import com.amazonaws.services.glue.util.GlueArgParser
import com.amazonaws.services.glue.util.Job
import com.amazonaws.services.glue.util.JsonOptions
import com.amazonaws.services.glue.DynamoDbDataSink
import org.apache.spark.SparkContext
import scala.collection.JavaConverters._


object GlueApp {

  def main(sysArgs: Array[String]): Unit = {
    val glueContext = new GlueContext(SparkContext.getOrCreate())
    val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB_NAME").toArray)
    Job.init(args("JOB_NAME"), glueContext, args.asJava)

    val dynamicFrame = glueContext.getCatalogSource(
        database = catalog_database,
        tableName = catalog_table_name,
        additionalOptions = JsonOptions(Map(
            "dynamodb.export" -> "ddb",
            "dynamodb.s3.bucket" -> s3_bucket,
            "dynamodb.s3.prefix" -> s3_bucket_prefix
        ))
    ).getDynamicFrame()
    print(dynamicFrame.getNumPartitions())
)
```

Simplificación del uso del JSON de exportación de DynamoDB

Las exportaciones de DynamoDB con el conector de exportación de DynamoDB de AWS Glue generan archivos JSON con estructuras anidadas específicas. Para obtener más información,

consulte Objetos de datos. AWS Glue proporciona una transformación de DynamicFrame, que puede desanidar estas estructuras y dejarlas con una forma más fácil de usar para aplicaciones posteriores.

La transformación se puede invocar de dos formas. Puede configurar la opción de conexión de `"dynamodb.simplifyDDBJson"` con el valor `"true"` cuando se llame a un método para leer desde DynamoDB. También puede llamar a la transformación como un método disponible de forma independiente en la biblioteca de AWS Glue.

Considere el siguiente esquema generado por una exportación a DynamoDB:

```
root
|-- Item: struct
|    |-- parentMap: struct
|    |    |-- M: struct
|    |    |    |-- childMap: struct
|    |    |    |    |-- M: struct
|    |    |    |    |    |-- appName: struct
|    |    |    |    |    |    |-- S: string
|    |    |    |    |    |-- packageName: struct
|    |    |    |    |    |    |-- S: string
|    |    |    |    |    |-- updatedAt: struct
|    |    |    |    |    |    |-- N: string
|    |-- strings: struct
|    |    |-- SS: array
|    |    |    |-- element: string
|    |-- numbers: struct
|    |    |-- NS: array
|    |    |    |-- element: string
|    |-- binaries: struct
|    |    |-- BS: array
|    |    |    |-- element: string
|    |-- isDDBJson: struct
|    |    |-- BOOL: boolean
|    |-- nullValue: struct
|    |    |-- NULL: boolean
```

La transformación de `simplifyDDBJson` simplificará esto de la siguiente forma:

```
root
|-- parentMap: struct
|    |-- childMap: struct
|    |    |-- appName: string
```

```
|    |    |-- packageName: string
|    |    |-- updatedAt: string
|-- strings: array
|    |-- element: string
|-- numbers: array
|    |-- element: string
|-- binaries: array
|    |-- element: string
|-- isDDBJson: boolean
|-- nullValue: null
```

> **Note**
>
> `simplifyDDBJson` solo está disponible en AWS Glue 3.0 o versiones posteriores. La transformación `unnestDDBJson` también está disponible para simplificar la exportación de JSON a DynamoDB. Alentamos a los usuarios a realizar la transición de `unnestDDBJson` a `simplifyDDBJson`.

Configuración del paralelismo en las operaciones de DynamoDB

Para mejorar el rendimiento, puede ajustar determinados parámetros disponibles para el conector DynamoDB. Su objetivo al ajustar los parámetros de paralelismo es maximizar el uso de los trabajadores de AWS Glue aprovisionados. Por lo tanto, si necesita más rendimiento, le recomendamos que amplíe su trabajo al aumentar el número de DPU.

Puede modificar el paralelismo en una operación de lectura de DynamoDB mediante el parámetro `dynamodb.splits` cuando utilice el conector ETL. Cuando lea con el conector de exportación, no es necesario configurar el número de divisiones del paralelismo del ejecutor de Spark. Puede modificar el paralelismo en una operación de escritura de DynamoDB con `dynamodb.output.numParallelTasks`.

Lectura con el conector ETL de DynamoDB

Le recomendamos que calcule `dynamodb.splits` en función del número máximo de trabajadores establecido en la configuración de su trabajo y del siguiente cálculo `numSlots`. Si se autoescala, la cantidad real de trabajadores disponibles puede cambiar por debajo de ese límite. Para obtener más información sobre cómo establecer el número máximo de trabajadores, consulte Número de trabajadores (`NumberOfWorkers`) en the section called “Configurar las propiedades de los trabajos de Spark”.

- `numExecutors = NumberOfWorkers - 1`

  Para contextualizar, un ejecutor está reservado para el controlador de Spark mientras que otros ejecutores procesan datos.
- `numSlotsPerExecutor =`

  AWS Glue 3.0 and later versions
  - 4 si `WorkerType` es `G.1X`
  - 8 si `WorkerType` es `G.2X`
  - 16 si `WorkerType` es `G.4X`
  - 32 si `WorkerType` es `G.8X`

  AWS Glue 2.0 and legacy versions
  - 8 si `WorkerType` es `G.1X`
  - 16 si `WorkerType` es `G.2X`
- `numSlots = numSlotsPerExecutor * numExecutors`

Le recomendamos que establezca `dynamodb.splits` al número de ranuras disponibles, `numSlots`.

Escritura en DynamoDB

El parámetro `dynamodb.output.numParallelTasks` se utiliza para determinar la WCU por tarea de Spark mediante el siguiente cálculo:

`permittedWcuPerTask = ( TableWCU * dynamodb.throughput.write.percent ) / dynamodb.output.numParallelTasks`

El escritor de DynamoDB funcionará mejor si la configuración representa con precisión el número de tareas de Spark que se escriben en DynamoDB. En algunos casos, es posible que necesite anular el cálculo predeterminado para mejorar el rendimiento de la escritura. Si no especifica este parámetro, el WCU permitido por tarea de Spark se calculará automáticamente mediante la siguiente fórmula:

- • `numPartitions = dynamicframe.getNumPartitions()`
  - `numSlots` (tal y como se definió anteriormente en esta sección)
  - `numParallelTasks = min(numPartitions, numSlots)`
- Ejemplo 1. DPU=10, tipo de empleado=estándar. La entrada DynamicFrame tiene 100 particiones RDD.

  - `numPartitions = 100`
  - `numExecutors = (10 - 1) * 2 - 1 = 17`
  - `numSlots = 4 * 17 = 68`
  - `numParallelTasks = min(100, 68) = 68`
- Ejemplo 2. DPU=10, tipo de empleado=estándar. La entrada DynamicFrame tiene 20 particiones RDD.
  - `numPartitions = 20`
  - `numExecutors = (10 - 1) * 2 - 1 = 17`
  - `numSlots = 4 * 17 = 68`
  - `numParallelTasks = min(20, 68) = 20`

> **Note**
>
> Los trabajos heredados de las versiones de AWS Glue, y los que utilizan trabajadores estándar, requieren diferentes métodos para calcular el número de ranuras. Si necesita ajustar el rendimiento de estos trabajos, le recomendamos que haga la transición a las versiones de AWS Glue compatibles.

Referencia de opciones de conexión de DynamoDB

Designa una conexión a Amazon DynamoDB.

Las opciones de conexión son distintas entre una conexión de origen y una conexión de receptor.

"connectionType": "dynamodb" con el conector de ETL como origen

Utilice las siguientes opciones de conexión con `"connectionType": "dynamodb"` como origen cuando use el conector de ETL de DynamoDB de AWS Glue:

- `"dynamodb.input.tableName"`: (obligatorio) la tabla de DynamoDB de la que se va a leer.
- `"dynamodb.throughput.read.percent"`: (opcional) porcentaje de unidades de capacidad de lectura (RCU) que se usará. El valor predeterminado se establece en "0,5". Los valores aceptables abarcan de "0,1" a "1,5", inclusive.
  - `0.5` representa la tasa de lectura predeterminada, es decir que AWS Glue intentará consumir la mitad de la capacidad de lectura de la tabla. Si usted aumenta el valor por encima de `0.5`, AWS

Glue incrementará la tasa de solicitudes; si reduce el valor por debajo de `0.5`, disminuirá la tasa de solicitudes de lectura. (La tasa de lectura real varía, según diversos factores tales como el hecho de que exista o no una distribución uniforme de claves en la tabla de DynamoDB).

- Cuando la tabla de DynamoDB está en modo bajo demanda, AWS Glue maneja la capacidad de lectura de la tabla como 40 000. Para exportar una tabla de gran extensión, recomendamos cambiar la tabla de DynamoDB al modo bajo demanda.

- `"dynamodb.splits"`: (opcional) define la cantidad de particiones en las que dividimos esta tabla de DynamoDB al leerla. El valor predeterminado se establece en "1". Los valores aceptables abarcan de "1" a "1,000,000", inclusive.

  1 representa que no hay paralelismo. Se recomienda especialmente que especifique un valor mayor para un mejor rendimiento mediante la fórmula siguiente. Para obtener más información sobre cómo configurar un valor de forma adecuada, consulte the section called “Paralelismo de DynamoDB”.
- `"dynamodb.sts.roleArn"`: (opcional) el ARN de rol de IAM que se asumirá para el acceso entre cuentas. Este parámetro se encuentra disponible en AWS Glue 1.0 o posterior.
- `"dynamodb.sts.roleSessionName"`: (opcional) nombre de sesión STS. El valor predeterminado se establece en “glue-dynamodb-read-sts-session”. Este parámetro se encuentra disponible en AWS Glue 1.0 o posterior.

“connectionType”: “dynamodb” con el conector de exportación de DynamoDB de AWS Glue como origen

Utilice las siguientes opciones de conexión con “connectionType”: “dynamodb” como origen cuando use el conector de exportación de DynamoDB de AWS Glue, que solo está disponible a partir de AWS Glue versión 2.0:

- `"dynamodb.export"`: (obligatorio) valor de cadena:
  - Si se configura como `ddb`, habilita el conector de exportación de DynamoDB de AWS Glue cuando se invoque una nueva `ExportTableToPointInTimeRequest` durante el trabajo de AWS Glue. Se generará una nueva exportación con la ubicación pasada desde `dynamodb.s3.bucket` y `dynamodb.s3.prefix`.
  - Si se configura como `s3`, habilita el conector de exportación de DynamoDB de AWS Glue, pero omite la creación de una nueva exportación de DynamoDB y, en su lugar, utiliza `dynamodb.s3.bucket` y `dynamodb.s3.prefix` como ubicación de Simple Storage Service (Amazon S3) de una exportación anterior de esa tabla.

- `"dynamodb.tableArn"`: (obligatorio) tabla de DynamoDB desde la que se debe leer.
- `"dynamodb.unnestDDBJson"`: (Opcional) Predeterminado: falso. Valores válidos: booleano. Si se configura como true, realiza una transformación no anidada de la estructura JSON de DynamoDB que está presente en las exportaciones. Es un error si se establecen `"dynamodb.unnestDDBJson"` y `"dynamodb.simplifyDDBJson"` como verdaderos al mismo tiempo. En AWS Glue 3.0 y versiones posteriores, le recomendamos que lo utilice `"dynamodb.simplifyDDBJson"` para mejorar el comportamiento al simplificar los tipos de mapas de DynamoDB. Para obtener más información, consulte the section called “Simplificación del uso del JSON de exportación de DynamoDB”.
- `"dynamodb.simplifyDDBJson"`: (Opcional) Predeterminado: falso. Valores válidos: booleano. Si se configura como verdadero, realiza una transformación para simplificar el esquema de JSON de DynamoDB que está presente en las exportaciones. Tiene el mismo propósito que la opción `"dynamodb.unnestDDBJson"`, pero proporciona un mejor soporte para los tipos de mapas de DynamoDB o incluso con los tipos de mapas anidados en su tabla de DynamoDB. Esta característica solo está disponible en AWS Glue 3.0 o versiones posteriores. Es un error si se establecen `"dynamodb.unnestDDBJson"` y `"dynamodb.simplifyDDBJson"` como verdaderos al mismo tiempo. Para obtener más información, consulte the section called “Simplificación del uso del JSON de exportación de DynamoDB”.
- `"dynamodb.s3.bucket"`: (opcional) indica la ubicación del bucket de Simple Storage Service (Amazon S3) en el que debe llevarse a cabo el proceso `ExportTableToPointInTime` de DynamoDB. El formato de archivo para la exportación es DynamoDB JSON.
  - `"dynamodb.s3.prefix"`: (opcional) indica la ubicación del prefijo de Simple Storage Service (Amazon S3) dentro del bucket de Simple Storage Service (Amazon S3) en el que se almacenarán las cargas `ExportTableToPointInTime` de DynamoDB. Si no se especifica `dynamodb.s3.prefix` ni `dynamodb.s3.bucket`, estos valores adoptarán de manera predeterminada la ubicación del directorio temporal especificada en la configuración del trabajo de AWS Glue. Para obtener más información, consulte Parámetros especiales utilizados por AWS Glue.
  - `"dynamodb.s3.bucketOwner"`: indica el propietario del bucket que se necesita para el acceso entre cuentas de Simple Storage Service (Amazon S3).
- `"dynamodb.sts.roleArn"`: (opcional) ARN del rol de IAM que se asumirá para el acceso entre cuentas o el acceso entre regiones para la tabla de DynamoDB. Nota: El mismo ARN del rol de IAM se utilizará para acceder a la ubicación de Simple Storage Service (Amazon S3) especificada para la solicitud `ExportTableToPointInTime`.

- `"dynamodb.sts.roleSessionName"`: (opcional) nombre de sesión STS. El valor predeterminado se establece en “glue-dynamodb-read-sts-session”.
- `"dynamodb.exportTime"` (Opcional) Valores válidos: cadenas que representan instantes de la norma ISO-8601. Un punto en el tiempo en el que debe realizarse la exportación.
- `"dynamodb.sts.region"`: (Obligatorio si se realiza una llamada entre regiones mediante un punto de conexión regional) La región que aloja la tabla de DynamoDB que desea leer.

“connectionType”: “dynamodb” con el conector de ETL como receptor

Utilice las siguientes opciones de conexión con `"connectionType": "dynamodb"` como receptor:

- `"dynamodb.output.tableName"`: (obligatorio) tabla de DynamoDB en la que se va a escribir.
- `"dynamodb.throughput.write.percent"`: (opcional) porcentaje de unidades de capacidad de escritura (WCU) que se usará. El valor predeterminado se establece en "0,5". Los valores aceptables abarcan de "0,1" a "1,5", inclusive.
  - `0.5` representa la tasa de escritura predeterminada, es decir que AWS Glue intentará consumir la mitad de la capacidad de escritura de la tabla. Si usted aumenta el valor por encima de 0,5, AWS Glue incrementará la tasa de solicitudes; si reduce el valor por debajo de 0,5, disminuirá la tasa de solicitudes de escritura. (La tasa de escritura real varía en función de diversos factores, tales como el hecho de que exista o no una distribución uniforme de claves en la tabla de DynamoDB).
  - Cuando la tabla de DynamoDB está en modo bajo demanda, AWS Glue maneja la capacidad de escritura de la tabla como `40000`. Para importar una tabla grande, recomendamos cambiar la tabla de DynamoDB al modo bajo demanda.
- `"dynamodb.output.numParallelTasks"`: (opcional) define el número de tareas paralelas que escriben en DynamoDB al mismo tiempo. Se utiliza para calcular WCU permisiva por tarea de Spark. En la mayoría de los casos, AWS Glue calculará un valor predeterminado razonable para este valor. Para obtener más información, consulte the section called “Paralelismo de DynamoDB”.
- `"dynamodb.output.retry"`: (opcional) define el número de reintentos que realizamos cuando hay una `ProvisionedThroughputExceededException` de DynamoDB. El valor predeterminado se establece en “10”.
- `"dynamodb.sts.roleArn"`: (opcional) el ARN de rol de IAM que se asumirá para el acceso entre cuentas.
- `"dynamodb.sts.roleSessionName"`: (opcional) nombre de sesión STS. El valor predeterminado se establece en “glue-dynamodb-write-sts-session”.

Acceso entre cuentas y entre regiones a tablas de DynamoDB

Los trabajos de ETL de AWS Glue soportan el acceso entre cuentas y entre regiones a tablas de DynamoDB. AWS Glue Los trabajos de ETL soportan tanto la lectura de datos de una tabla de DynamoDB de otra cuenta AWS, como la escritura de datos de una tabla de DynamoDB de otra cuenta AWS. AWS Glue también soporta la lectura de una tabla de DynamoDB en otra región y la escritura en una tabla de DynamoDB en otra región. En esta sección se proporcionan instrucciones sobre cómo configurar el acceso y se proporciona un script de ejemplo.

Los procedimientos de esta sección hacen referencia a un tutorial de IAM para crear un rol de IAM y conceder acceso al rol. El tutorial también explica cómo asumir un rol, pero aquí usará un script de trabajo para asumir el rol en AWS Glue. Este tutorial también contiene información acerca de las prácticas generales entre cuentas. Para obtener más información, consulte Tutorial: delegar el acceso entre cuentas de AWS mediante roles de IAM en la Guía del usuario de IAM.

Crear un rol

Siga el paso 1 en el tutorial para crear un rol de IAM en la cuenta A. Al definir los permisos del rol, puede optar por asociar políticas existentes como `AmazonDynamoDBReadOnlyAccess` o `AmazonDynamoDBFullAccess` para permitir que el rol lea y escriba DynamoDB. En el siguiente ejemplo, se muestra cómo crear un rol llamado `DynamoDBCrossAccessRole`, con la política de permisos `AmazonDynamoDBFullAccess`.

Conceder acceso al rol

Siga el paso 2 en el tutorial en la Guía del usuario de IAM para permitir que la cuenta B cambie al rol recién creado. En el siguiente ejemplo, se crea una nueva política con la siguiente instrucción:

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": "sts:AssumeRole",
        "Resource": "<DynamoDBCrossAccessRole's ARN>"
    }
}
```

A continuación, puede asociar esta política al grupo/rol o usuario que desea utilizar para acceder a DynamoDB.

Asumir el rol en el script de trabajo de AWS Glue

Ahora, puede iniciar sesión en la cuenta B y crear un trabajo de AWS Glue. Para crear un trabajo, consulte las instrucciones en Configurar las propiedades de los trabajos de Spark en AWS Glue.

En el script de trabajo, debe usar el parámetro `dynamodb.sts.roleArn` para asumir el rol `DynamoDBCrossAccessRole`. Asumir este rol le permite obtener las credenciales temporales que deben utilizarse para acceder a DynamoDB en la cuenta B. Revise estos scripts de ejemplo.

Para una lectura entre cuentas entre regiones (conector ETL):

```
import sys
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job
from awsglue.utils import getResolvedOptions

args = getResolvedOptions(sys.argv, ["JOB_NAME"])
glue_context= GlueContext(SparkContext.getOrCreate())
job = Job(glue_context)
job.init(args["JOB_NAME"], args)

dyf = glue_context.create_dynamic_frame_from_options(
    connection_type="dynamodb",
    connection_options={
    "dynamodb.region": "us-east-1",
    "dynamodb.input.tableName": "test_source",
    "dynamodb.sts.roleArn": "<DynamoDBCrossAccessRole's ARN>"
    }
)
dyf.show()
job.commit()
```

Para una lectura entre cuentas entre regiones (conector ELT):

```
import sys
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job
from awsglue.utils import getResolvedOptions

args = getResolvedOptions(sys.argv, ["JOB_NAME"])
glue_context= GlueContext(SparkContext.getOrCreate())
```

```
job = Job(glue_context)
job.init(args["JOB_NAME"], args)

dyf = glue_context.create_dynamic_frame_from_options(
    connection_type="dynamodb",
    connection_options={
        "dynamodb.export": "ddb",
        "dynamodb.tableArn": "<test_source ARN>",
        "dynamodb.sts.roleArn": "<DynamoDBCrossAccessRole's ARN>"
    }
)
dyf.show()
job.commit()
```

Para una lectura y escritura entre cuentas entre regiones:

```
import sys
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job
from awsglue.utils import getResolvedOptions

args = getResolvedOptions(sys.argv, ["JOB_NAME"])
glue_context= GlueContext(SparkContext.getOrCreate())
job = Job(glue_context)
job.init(args["JOB_NAME"], args)

dyf = glue_context.create_dynamic_frame_from_options(
    connection_type="dynamodb",
    connection_options={
        "dynamodb.region": "us-east-1",
        "dynamodb.input.tableName": "test_source"
    }
)
dyf.show()

glue_context.write_dynamic_frame_from_options(
    frame=dyf,
    connection_type="dynamodb",
    connection_options={
        "dynamodb.region": "us-west-2",
        "dynamodb.output.tableName": "test_sink",
        "dynamodb.sts.roleArn": "<DynamoDBCrossAccessRole's ARN>"
```

```
    }
)

job.commit()
```

Conexión de Kinesis

Puede leer y escribir en Amazon Kinesis Data Streams mediante la información almacenada en una tabla de Catálogo de datos o al brindar información para acceder directamente al flujo de datos. Puede leer la información de Kinesis en un Spark y DataFrame, después, convertirla en un Glue AWS . DynamicFrame Puede DynamicFrames escribir en Kinesis en formato JSON. Si accede directamente a la secuencia de datos, utilice estas opciones para proporcionar información sobre cómo acceder a la secuencia de datos.

Si utiliza `getCatalogSource` o `create_data_frame_from_catalog` para consumir registros de una fuente de streaming de Kinesis, el trabajo tiene la base de datos de Data Catalog y la información del nombre de la tabla, y puede utilizarla para obtener algunos parámetros básicos para la lectura de la fuente de streaming de Kinesis. Si utiliza `getSource`, `getSourceWithFormat`, `createDataFrameFromOptions` o `create_data_frame_from_options` debe especificar estos parámetros básicos mediante las opciones de conexión descritas aquí.

Puede especificar las opciones de conexión para Kinesis al utilizar los siguientes argumentos para los métodos especificados en la clase `GlueContext`.

- Scala
  - `connectionOptions`: se debe utilizar con `getSource`, `createDataFrameFromOptions` y `getSink`
  - `additionalOptions`: se debe utilizar con `getCatalogSource`, `getCatalogSink`
  - `options`: se debe utilizar con `getSourceWithFormat`, `getSinkWithFormat`
- Python
  - `connection_options`: se debe utilizar con `create_data_frame_from_options`, `write_dynamic_frame_from_options`
  - `additional_options`: se debe utilizar con `create_data_frame_from_catalog`, `write_dynamic_frame_from_catalog`
  - `options`: se debe utilizar con `getSource`, `getSink`

Para obtener notas y restricciones sobre los trabajos de ETL de Streaming, consulte the section called “Notas y restricciones de ETL de streaming”.

Configurar Kinesis

Para conectarse a una transmisión de datos de Kinesis en un trabajo de AWS Glue Spark, necesitará algunos requisitos previos:

- Si está leyendo, el trabajo de AWS Glue debe tener permisos de IAM de nivel de acceso de lectura para la transmisión de datos de Kinesis.
- Si está escribiendo, el trabajo de AWS Glue debe tener permisos de IAM de nivel de acceso de escritura para la transmisión de datos de Kinesis.

En algunos casos, tendrá que configurar requisitos previos adicionales:

- Si su trabajo de AWS Glue está configurado con conexiones de red adicionales (normalmente para conectarse a otros conjuntos de datos) y una de esas conexiones proporciona opciones de red de Amazon VPC, esto indicará que su trabajo se comunique a través de Amazon VPC. En este caso, también tendrá que configurar el flujo de datos de Kinesis para que se comunique a través de la VPC de Amazon. Puede hacerlo mediante la creación de un punto de conexión de VPC de tipo interfaz entre la VPC de Amazon y el flujo de datos de Kinesis. Para obtener más información, consulte Uso de Kinesis de Amazon Kinesis Data Streams con puntos de conexión de VPC de interfaz.
- Al especificar Amazon Kinesis Data Streams en otra cuenta, debe configurar los roles y las políticas para permitir el acceso entre cuentas. Para obtener más información, consulte Ejemplo: leer desde un flujo de Kinesis en una cuenta diferente.

Para obtener más información sobre los requisitos previos del trabajo de ETL de Streaming, consulte the section called “Trabajos ETL de streaming”.

Ejemplo: lectura de transmisiones desde Kinesis

Ejemplo: lectura de transmisiones desde Kinesis

Se utiliza junto con the section called “forEachBatch”.

Ejemplo de origen de streaming de Amazon Kinesis:

```
kinesis_options =
```

```
   { "streamARN": "arn:aws:kinesis:us-east-2:777788889999:stream/fromOptionsStream",
     "startingPosition": "TRIM_HORIZON",
     "inferSchema": "true",
     "classification": "json"
   }
data_frame_datasource0 =
 glueContext.create_data_frame.from_options(connection_type="kinesis",
 connection_options=kinesis_options)
```

Ejemplo: escribir en flujos de Kinesis

Ejemplo: lectura de transmisiones desde Kinesis

Se utiliza junto con the section called “forEachBatch”.

Ejemplo de origen de streaming de Amazon Kinesis:

```
kinesis_options =
   { "streamARN": "arn:aws:kinesis:us-east-2:777788889999:stream/fromOptionsStream",
     "startingPosition": "TRIM_HORIZON",
     "inferSchema": "true",
     "classification": "json"
   }
data_frame_datasource0 =
 glueContext.create_data_frame.from_options(connection_type="kinesis",
 connection_options=kinesis_options)
```

Referencia de opciones de conexión de Kinesis

Designa opciones de conexión para Amazon Kinesis Data Streams.

Utilice las siguientes opciones de conexión para los orígenes de datos de streaming de Kinesis:

- `"streamARN"`: (Obligatorio) Se utiliza para leer/escribir. El ARN de flujo de datos de Kinesis.
- `"classification"`: (Obligatorio para lectura) Se utiliza para leer. El formato de archivo utilizado por los datos del registro. Obligatorio a menos que se proporcione a través del catálogo de datos.
- `"streamName"`: (Opcional) Se usa para leer. Nombre de un flujo de datos de Kinesis para leer. Utilizado con `endpointUrl`.
- `"endpointUrl"`: (Opcional) Se usa para leer. Predeterminado: “https://kinesis.us-east-1.amazonaws.com”. El AWS punto final de la transmisión de Kinesis. No es necesario cambiar esto a menos que se conecte a una región especial.

- `"partitionKey"`: (Opcional) Se usa para escribir. La clave de partición de Kinesis que se utiliza al producir registros.
- `"delimiter"`: (Opcional) Se usa para leer. El separador de valores que se utiliza cuando `classification` es CSV. El valor predeterminado es “`,`”.
- `"startingPosition"`: (Opcional) Se usa para leer. La posición inicial en el flujo de datos de Kinesis para leer los datos. Los valores posibles son `"latest"`, `"trim_horizon"`, `"earliest"` o una cadena de marca de tiempo en formato UTC en el patrón `yyyy-mm-ddTHH:MM:SSZ` (donde `Z` representa un desplazamiento de zona horaria UTC con un +/-. Por ejemplo, “04-04-2023 T 08:00:00-04:00”). El valor predeterminado es `"latest"`. Nota: la cadena Timestamp en formato UTC solo `"startingPosition"` es compatible con la versión 4.0 o posterior de AWS Glue.
- `"failOnDataLoss"`: (Opcional) No se realizará el trabajo si falta o ha caducado alguna partición activa. El valor predeterminado es `"false"`.
- `"awsSTSRoleARN"`: (Opcional) Se usa para escribir/leer. El nombre del recurso de Amazon (ARN) de la función que se va a asumir mediante AWS Security Token Service (AWS STS). Este rol debe tener permisos para describir o leer operaciones de registros del flujo de datos de Kinesis. Debe utilizar este parámetro para acceder a un flujo de datos de otra cuenta. Se utiliza junto con `"awsSTSSessionName"`.
- `"awsSTSSessionName"`: (Opcional) Se usa para escribir/leer. Un identificador para la sesión que asume el rol mediante AWS STS. Debe utilizar este parámetro para acceder a un flujo de datos de otra cuenta. Se utiliza junto con `"awsSTSRoleARN"`.
- `"awsSTSEndpoint"`: (Opcional) El AWS STS punto final que se utilizará al conectarse a Kinesis con un rol asumido. Esto permite usar el AWS STS punto final regional en una VPC, lo que no es posible con el punto final global predeterminado.
- `"maxFetchTimeInMs"`: (Opcional) Se usa para leer. El tiempo máximo que le tomó al ejecutor del trabajo leer los registros del lote actual en el flujo de datos de Kinesis, especificado en milisegundos (ms). Pueden realizarse varias llamadas a la API de `GetRecords` durante este tiempo. El valor predeterminado es `1000`.
- `"maxFetchRecordsPerShard"`: (Opcional) Se usa para leer. El número máximo de registros que se recuperará por partición en el flujo de datos de Kinesis por microlote. Nota: El cliente puede exceder este límite si el trabajo de streaming ya leyó registros adicionales de Kinesis (en la misma llamada de obtención de registros). Si `maxFetchRecordsPerShard` necesita ser preciso, entonces necesita ser un múltiplo de `maxRecordPerRead`. El valor predeterminado es `100000`.

- `"maxRecordPerRead"`: (Opcional) Se usa para leer. El número máximo de registros que se recuperará del flujo de datos de Kinesis en cada operación `getRecords`. El valor predeterminado es `10000`.
- `"addIdleTimeBetweenReads"`: (Opcional) Se usa para leer. Agrega un retardo de tiempo entre dos operaciones `getRecords` consecutivas. El valor predeterminado es `"False"`. Esta opción sólo se puede configurar para Glue versión 2.0 y superior.
- `"idleTimeBetweenReadsInMs"`: (Opcional) Se usa para leer. El tiempo mínimo de retraso entre dos operaciones `getRecords` consecutivas, especificado en ms. El valor predeterminado es `1000`. Esta opción sólo se puede configurar para Glue versión 2.0 y superior.
- `"describeShardInterval"`: (Opcional) Se usa para leer. El intervalo mínimo de tiempo entre dos llamadas a la API `ListShards` para que su script considere cambios en los fragmentos. Para obtener más información, consulte Estrategias para cambios en los fragmentos en la Guía para desarrolladores de Amazon Kinesis Data Streams. El valor predeterminado es `1s`.
- `"numRetries"`: (Opcional) Se usa para leer. El número máximo de reintentos para las solicitudes de la API de Kinesis Data Streams. El valor predeterminado es `3`.
- `"retryIntervalMs"`: (Opcional) Se usa para leer. El periodo de enfriamiento (especificado en ms) antes de volver a intentar la llamada a la API de Kinesis Data Streams. El valor predeterminado es `1000`.
- `"maxRetryIntervalMs"`: (Opcional) Se usa para leer. El periodo de enfriamiento máximo (especificado en ms) entre dos intentos de llamada a la API de Kinesis Data Streams. El valor predeterminado es `10000`.
- `"avoidEmptyBatches"`: (Opcional) Se usa para leer. Evita crear un trabajo de microlotes vacío al comprobar si hay datos no leídos en el flujo de datos de Kinesis antes de que se inicie el lote. El valor predeterminado es `"False"`.
- `"schema"`: (Obligatorio cuando inferSchema se establece en false) Se utiliza para leer. El esquema que se utilizará para procesar la carga útil. Si la clasificación es `avro`, el esquema proporcionado debe estar en el formato de esquema Avro. Si la clasificación no es `avro`, el esquema proporcionado debe estar en el formato de esquema DDL.

  A continuación, se muestran algunos ejemplos de esquemas.

  Example in DDL schema format

  ```
  `column1` INT, `column2` STRING , `column3` FLOAT
  ```

Example in Avro schema format

```
{
  "type":"array",
  "items":
  {
    "type":"record",
    "name":"test",
    "fields":
    [
      {
        "name":"_id",
        "type":"string"
      },
      {
        "name":"index",
        "type":
        [
          "int",
          "string",
          "float"
        ]
      }
    ]
  }
}
```

- `"inferSchema"`: (Opcional) Se usa para leer. El valor predeterminado es "false". Si se establece en “true”, el esquema se detectará durante el tiempo de ejecución desde la carga dentro de `foreachbatch`.
- `"avroSchema"`: (Obsoleto) Se usa para leer. Parámetro utilizado para especificar un esquema de datos Avro cuando se utiliza el formato Avro. Este parámetro se ha quedado obsoleto. Utilice el parámetro `schema`.
- `"addRecordTimestamp"`: (Opcional) Se usa para leer. Cuando esta opción se establece en “true”, la salida de datos contendrá una columna adicional denominada “__src_timestamp” que indica la hora en la que el flujo recibió el registro correspondiente. El valor predeterminado es "false". Esta opción es compatible con la versión 4.0 o posterior de AWS Glue.
- `"emitConsumerLagMetrics"`: (Opcional) Se usa para leer. Si la opción se establece en «true», para cada lote, emitirá las métricas correspondientes al período comprendido entre el registro más antiguo recibido por la transmisión y el momento en que llegue AWS Glue a

CloudWatch él. El nombre de la métrica es «glue.driver.streaming». maxConsumerLagInMs». El valor predeterminado es "false". Esta opción es compatible con la versión 4.0 o posterior de AWS Glue.

- `"fanoutConsumerARN"`: (Opcional) Se usa para leer. El ARN de un consumidor de un flujo de Kinesis para el flujo especificado en `streamARN`. Se utiliza para habilitar el modo de distribución mejorada para la conexión de Kinesis. Para obtener más información sobre cómo consumir una transmisión de Kinesis con una distribución mejorada, consulte the section called “Uso de una distribución mejorada en los trabajos de streaming de Kinesis”.
- `"recordMaxBufferedTime"`: (Opcional) Se usa para escribir. Predeterminado: 1000 (ms). Tiempo máximo que un registro permanece almacenado en búfer mientras espera a ser escrito.
- `"aggregationEnabled"`: (Opcional) Se usa para escribir. Valor predeterminado: verdadero. Especifica si los registros deben agregarse antes de enviarlos a Kinesis.
- `"aggregationMaxSize"`: (Opcional) Se usa para escribir. Predeterminado: 51 200 (bytes). Si un registro supera este límite, omitirá el agregador. Nota: Kinesis impone un límite de 50 KB en el tamaño del registro. Si lo establece por encima de 50 KB, Kinesis rechazará los registros de gran tamaño.
- `"aggregationMaxCount"`: (Opcional) Se usa para escribir. Predeterminado: 4294967295. Número máximo de elementos a empaquetar en un registro agregado.
- `"producerRateLimit"`: (Opcional) Se usa para escribir. Predeterminado: 150 (%). Limita el rendimiento por partición enviado desde un solo productor (por ejemplo, su trabajo), como porcentaje del límite de backend.
- `"collectionMaxCount"`: (Opcional) Se usa para escribir. Predeterminado: 500. Número máximo de artículos para incluir en una PutRecords solicitud.
- `"collectionMaxSize"`: (Opcional) Se usa para escribir. Predeterminado: 5 242 880 (bytes). Cantidad máxima de datos que se pueden enviar con una PutRecords solicitud.

Uso de una distribución mejorada en los trabajos de streaming de Kinesis

Un consumidor con una distribución mejorada puede recibir los registros de una transmisión de Kinesis con un rendimiento dedicado que puede ser superior al de los consumidores habituales. Esto se logra mediante la optimización del protocolo de transferencia utilizado para proporcionar datos a un consumidor de Kinesis, como su trabajo. Para obtener más información sobre Kinesis Enhanced Fan-Out, consulte la documentación de Kinesis.

En el modo de distribución mejorada, las opciones de conexión `maxRecordPerRead` y `idleTimeBetweenReadsInMs` ya no se aplican, ya que esos parámetros no se pueden configurar cuando se utiliza la apertura de distribución mejorada. Las opciones de configuración para los reintentos funcionan como se describe.

Utilice los siguientes procedimientos para habilitar y deshabilitar la distribución mejorada para su trabajo de streaming. Debe registrar un consumidor de transmisión para cada trabajo que vaya a consumir datos de su transmisión.

Para habilitar un consumo de distribución mejorada en su trabajo:

1. Registre un consumidor de transmisión para su trabajo mediante la API de Kinesis. Siga las instrucciones para registrar a un consumidor con una distribución mejorada mediante la API de Kinesis Data Streams de la documentación de Kinesis. Solo tendrá que seguir el primer paso: llamar a RegisterStreamConsumer. Su solicitud debe devolver un ARN, *`consumerARN`*.
2. Establezca la opción de conexión `fanoutConsumerARN` en *`consumerARN`* en los argumentos del método de conexión.
3. Reinicie el trabajo.

Para deshabilitar un consumo de distribución mejorada en su trabajo:

1. Elimine la opción de conexión `fanoutConsumerARN` de su método de llamada.
2. Reinicie el trabajo.
3. Siga las instrucciones para anular el registro de un consumidor en la documentación de Kinesis. Estas instrucciones se aplican a la consola, pero también se pueden obtener a través de la API de Kinesis. Para obtener más información sobre la cancelación del registro de consumidores de transmisión a través de la API de Kinesis, consulte DeregisterStreamConsumer en la documentación de Kinesis.

Conexiones de Amazon S3

Puede usar Glue AWS for Spark para leer y escribir archivos en Amazon S3. AWS Glue for Spark admite muchos formatos de datos habituales que se almacenan en Amazon S3 de forma inmediata, como CSV, Avro, JSON, Orc y Parquet. Para obtener más información acerca de los formatos de datos admitidos, consulte the section called “Opciones de formato de datos”. Cada formato de datos puede admitir diferentes características de Glue AWS. Consulte la página correspondiente a su formato de datos para conocer los detalles de la compatibilidad con las características. Además,

puede leer y escribir archivos versionados almacenados en los marcos de lago de datos de Hudi, Iceberg y Delta Lake. Para obtener más información sobre los marcos de lago de datos, consulte the section called “Marcos de lagos de datos”.

Con Glue AWS, puede particionar sus objetos de Amazon S3 en una estructura de carpetas mientras escribe y, a continuación, recuperarlos mediante una partición para mejorar el rendimiento mediante una configuración sencilla. También puede establecer la configuración para agrupar archivos pequeños al transformar sus datos para mejorar el rendimiento. Puede leer, escribir `bzip2` y archivar `gzip` en Amazon S3.

Temas

- Configuración de las conexiones S3
- Referencia de opción de conexión de Amazon S3
- Sintaxis de conexión obsoleta para los formatos de datos
- Exclusión de clases de almacenamiento de Amazon S3
- Administración de particiones para la salida de ETL en AWS Glue
- Lectura de archivos de entrada en grupos más grandes
- Puntos de enlace de Amazon VPC para Amazon S3

Configuración de las conexiones S3

Para conectarse a Amazon S3 en un trabajo de Glue with Spark AWS, necesitará algunos requisitos previos:

- El trabajo de Glue AWS debe tener permisos de IAM para los buckets de Amazon S3 correspondientes.

En algunos casos, tendrá que configurar requisitos previos adicionales:

- Al configurar el acceso entre cuentas, los controles de acceso adecuados en el bucket de Amazon S3.
- Por motivos de seguridad, puede optar por enrutar sus solicitudes de Amazon S3 a través de una VPC de Amazon. Este enfoque puede conllevar problemas de ancho de banda y disponibilidad. Para obtener más información, consulte the section called “Puntos de enlace de Amazon VPC para Amazon S3”.

Referencia de opción de conexión de Amazon S3

Designa una conexión a Amazon S3.

Dado que Amazon S3 administra archivos en lugar de tablas, además de especificar las propiedades de conexión que se proporcionan en este documento, tendrá que especificar una configuración adicional sobre su tipo de archivo. Puede especificar esta información mediante las opciones de formato de datos. Para obtener más información sobre las opciones de formato, consulte the section called “Opciones de formato de datos”. También puede especificar esta información integrándola con el catálogo de datos de Glue AWS.

Para ver un ejemplo de la distinción entre las opciones de conexión y las opciones de formato, considere la forma en que el método the section called “create_dynamic_frame_from_options” toma `connection_type`, `connection_options`, `format` y `format_options`. En esta sección se analizan específicamente los parámetros proporcionados a `connection_options`.

Utilice las siguientes opciones de conexión con `"connectionType": "s3"`:

- `"paths"`: (obligatorio) una lista de las rutas de Amazon S3 desde las que se va a leer.
- `"exclusions"`: (opcional) cadena que contiene una lista JSON de patrones glob de estilo Unix para excluir. Por ejemplo, `"[\"**.pdf\"]"` excluye todos los archivos PDF. Para obtener más información acerca de la sintaxis glob que admite AWS Glue, consulte Incluir y excluir patrones.
- `"compressionType"`: o `"compression"`: (opcional) especifica la forma en que los datos se comprimen. Use `"compressionType"` para orígenes de Amazon S3 y `"compression"` para destinos de Amazon S3. Po lo general no es necesario si los datos tienen una extensión de archivo estándar. Los posibles valores son `"gzip"` y `"bzip2"`). Es posible que se admitan formatos de compresión adicionales para formatos específicos. Consulte la página correspondiente a su formato de datos para conocer los detalles de la compatibilidad con las características.
- `"groupFiles"`: (opcional) la agrupación de archivos se habilita de forma predeterminada cuando la entrada contiene más de 50 000 archivos. Para habilitar las agrupaciones con menos de 50 000 archivos, establezca este parámetro en `"inPartition"`. Para deshabilitar las agrupaciones con más de 50 000 archivos, establezca este parámetro en `"none"`.
- `"groupSize"`: (opcional) tamaño del grupo de destino en bytes. El valor predeterminado se calcula en función del tamaño de los datos de entrada y el tamaño de su clúster. Cuando hay menos de 50 000 archivos de entrada, `"groupFiles"` debe establecerse en `"inPartition"` para que este valor surta efecto.
- `"recurse"`: (opcional) si se establece en verdadero, lee recursivamente archivos en todos los subdirectorios de las rutas especificadas.

- `"maxBand"`: (opcional, avanzada) esta opción controla la duración en milisegundos después de la que es probable que el listado de `s3` sea coherente. Se realiza un seguimiento de los archivos cuyas marcas de tiempo de modificación estén comprendidas en los últimos milisegundos de `maxBand`, en especial cuando se utilizan `JobBookmarks` para obtener coherencia eventual de Amazon S3. La mayoría de los usuarios no tienen que establecer esta opción. El valor predeterminado es 900 000 milisegundos, o 15 minutos.
- `"maxFilesInBand"`: (opcional, avanzada) esta opción especifica el número máximo de archivos que deben guardarse desde los últimos segundos de `maxBand`. Si se supera este número, los archivos adicionales se omiten y solo se procesarán en la siguiente ejecución del flujo de trabajo. La mayoría de los usuarios no tienen que establecer esta opción.
- `"isFailFast"`: (opcional) esta opción determina si un trabajo de ETL de AWS Glue arroja excepciones de análisis del lector. Si se establece en `true`, los trabajos fallan rápidamente si cuatro reintentos de la tarea Spark no pueden analizar los datos en forma correcta.
- `"catalogPartitionPredicate"`: (Opcional) Se usa para leer. El contenido de una cláusula SQL `WHERE`. Se utiliza al leer tablas del catálogo de datos con una gran cantidad de particiones. Recupera las particiones coincidentes de los índices del catálogo de datos. Se utiliza con `push_down_predicate`, una opción del método the section called “create_dynamic_frame_from_catalog” (y otros métodos similares). Para obtener más información, consulte the section called “Predicados de particiones de catálogo”.
- `"partitionKeys"`: (Opcional) Se usa para escribir. Matriz de cadenas de etiquetas de columnas. AWS Glue particionará los datos según lo especificado en esta configuración. Para obtener más información, consulte the section called “Escritura de particiones”.
- `"excludeStorageClasses"`: (Opcional) Se usa para leer. Matriz de cadenas que especifican las clases de almacenamiento de Amazon S3. AWS Glue excluirá los objetos de Amazon S3 en función de esta configuración. Para obtener más información, consulte the section called “Exclusión de clases de almacenamiento de Amazon S3”.

Sintaxis de conexión obsoleta para los formatos de datos

Se puede acceder a determinados formatos de datos mediante una sintaxis de tipo de conexión específica. Esta sintaxis está obsoleta. Recomendamos que especifique los formatos mediante la utilización del tipo de conexión `s3` y las opciones de formato que se proporcionan en the section called “Opciones de formato de datos” en su lugar.

"connectionType": "Orc"

Designa una conexión a archivos almacenados en Amazon S3 en formato de archivo Apache Hive Optimized Row Columnar (ORC).

Utilice las siguientes opciones de conexión con `"connectionType": "orc"`:

- `paths`: (obligatorio) una lista de las rutas de Amazon S3 desde las que se va a leer.
- (Otros pares de nombre/valor de la opción): cualquier opción adicional, incluidas las opciones de formato, se pasa directamente a SparkSQL `DataSource`.

"connectionType": "parquet"

Designa una conexión a archivos almacenados en Amazon S3 con el formato de archivo Apache Parquet.

Utilice las siguientes opciones de conexión con `"connectionType": "parquet"`:

- `paths`: (obligatorio) una lista de las rutas de Amazon S3 desde las que se va a leer.
- (Otros pares de nombre/valor de la opción): cualquier opción adicional, incluidas las opciones de formato, se pasa directamente a SparkSQL `DataSource`.

Exclusión de clases de almacenamiento de Amazon S3

Si ejecuta trabajos de ETL de AWS Glue que leen archivos o particiones desde Amazon Simple Storage Service (Amazon S3), puede excluir algunos tipos de clases de almacenamiento de Amazon S3.

Las siguientes clases de almacenamiento están disponibles en Amazon S3:

- `STANDARD`: para el almacenamiento de uso general de los datos a los que se accede con frecuencia.
- `INTELLIGENT_TIERING`: para datos de larga duración con patrones de acceso cambiantes o desconocidos.
- `STANDARD_IA` y `ONEZONE_IA`: para datos de larga duración, pero a los que se accede con menor frecuencia.
- `GLACIER`, `DEEP_ARCHIVE` y `REDUCED_REDUNDANCY`: para archivado a largo plazo y conservación digital.

Para obtener más información, consulte Clases de almacenamiento de Amazon S3 en la Guía para desarrolladores de Amazon S3.

Los ejemplos de esta sección muestran cómo excluir las clases de almacenamiento `DEEP_ARCHIVE` y `GLACIER`. Estas clases le permiten enumerar archivos, pero no le permiten leer los archivos a menos que se restauren. (Para obtener más información, consulte Restaurar objetos archivados en la Guía para desarrolladores de Amazon S3).

Mediante el uso de exclusiones de clase de almacenamiento, puede asegurarse de que sus trabajos de AWS Glue funcionarán en tablas que tengan particiones en estas capas de clase de almacenamiento. Sin exclusiones, los trabajos que leen datos de estas capas generan el siguiente error: AmazonS3Exception: The operation is not valid for the object's storage class (La operación no es válida para la clase de almacenamiento del objeto).

Existen diferentes formas de filtrar las clases de almacenamiento de Amazon S3 en AWS Glue.

Temas

- Exclusión de clases de almacenamiento de Amazon S3 al crear un marco dinámico
- Exclusión de clases de almacenamiento de Amazon S3 en una tabla de Data Catalog

Exclusión de clases de almacenamiento de Amazon S3 al crear un marco dinámico

Para excluir las clases de almacenamiento de Amazon S3 al crear un marco dinámico, utilice `excludeStorageClasses` en `additionalOptions`. AWS Glue utiliza automáticamente su propia implementación `Lister` de Amazon S3 para enumerar y excluir los archivos correspondientes a las clases de almacenamiento especificadas.

Los siguientes ejemplos de Python y Scala muestran cómo excluir las clases de almacenamiento `GLACIER` y `DEEP_ARCHIVE` al crear un marco dinámico.

Ejemplo de Python:

```
glueContext.create_dynamic_frame.from_catalog(
    database = "my_database",
    tableName = "my_table_name",
    redshift_tmp_dir = "",
    transformation_ctx = "my_transformation_context",
    additional_options = {
        "excludeStorageClasses" : ["GLACIER", "DEEP_ARCHIVE"]
    }
```

```
)
```

Ejemplo de Scala:

```
val* *df = glueContext.getCatalogSource(
    nameSpace, tableName, "", "my_transformation_context",
    additionalOptions = JsonOptions(
        Map("excludeStorageClasses" -> List("GLACIER", "DEEP_ARCHIVE"))
    )
).getDynamicFrame()
```

Exclusión de clases de almacenamiento de Amazon S3 en una tabla de Data Catalog

Puede especificar exclusiones de clase de almacenamiento que utilizará un trabajo de ETL de AWS Glue como parámetro de tabla en el Data Catalog de AWS Glue. Puede incluir este parámetro en la operación `CreateTable` mediante la AWS Command Line Interface (AWS CLI) o mediante programación con la API. Para obtener más información, consulte Estructura Table y CreateTable.

También puede especificar clases de almacenamiento excluidas en la consola de AWS Glue.

Para excluir clases de almacenamiento de Amazon S3 (consola)

1. Inicie sesión en la AWS Management Console y abra la consola de AWS Glue en https://console.aws.amazon.com/glue/.
2. En el panel de navegación de la izquierda, elija Tables (Tablas).
3. Elija el nombre de la tabla en la lista y, a continuación, elija Edit table (Editar tabla).
4. En Table properties (Propiedades de tabla), añada **`excludeStorageClasses`** como una clave y **`[\"GLACIER\",\"DEEP_ARCHIVE\"]`** como un valor.
5. Seleccione Aplicar.

Administración de particiones para la salida de ETL en AWS Glue

La creación de particiones es una importante técnica para organizar conjuntos de datos de manera que se puedan consultar de forma eficaz. Organiza los datos en una estructura de directorios jerárquica en función de los valores diferenciados de una o más columnas.

Por ejemplo, puede decidir particionar los registros de aplicación en Amazon Simple Storage Service (Amazon S3) por fecha, con desglose por año, mes y día. Los archivos que corresponden a los datos de un solo día se colocan con un prefijo como `s3://my_bucket/logs/year=2018/month=01/`

`day=23/`. Los sistemas como Amazon Athena, Amazon Redshift Spectrum, y ahora AWS Glue, pueden utilizar estas particiones para filtrar datos por valor de partición sin tener que leer todos los datos subyacentes desde Amazon S3.

Los rastreadores no solo infieren los tipos de archivo y esquemas, sino que también identifican automáticamente la estructura de particiones del conjunto de datos cuando rellenan AWS Glue Data Catalog. Las columnas de partición resultantes están disponibles para consultarlas en trabajos de ETL de AWS Glue o motores de consulta como Amazon Athena.

Después de rastrear una tabla, puede ver las particiones que creó el rastreador. En la consola de AWS Glue, seleccione Tables (Tablas) en el panel de navegación de la izquierda. Elija la tabla creada por el rastreador y, a continuación, elija View Partitions (Ver Particiones).

En el caso de las rutas con particiones de tipo Apache Hive en el estilo `key=val`, los rastreadores rellenan automáticamente el nombre de columna con el nombre de clave. De lo contrario, utiliza nombres predeterminados como `partition_0`, `partition_1` y así sucesivamente. Puede cambiar los nombres predeterminados de la consola. Para ello, navegue hasta la tabla. Compruebe si existen índices en la pestaña Indexes (Índices). Si ese es el caso, debe eliminarlos para continuar (puede volver a crearlos con los nuevos nombres de columna más adelante). A continuación, seleccione Editar esquema y modifique allí los nombres de las columnas de partición.

En los scripts de ETL, puede filtrar por las columnas de partición. Dado que la información de partición se almacena en el Data Catalog, utilice llamadas a la API `from_catalog` para incluir las columnas de partición en el `DynamicFrame`. Por ejemplo, utilice `create_dynamic_frame.from_catalog` en lugar de `create_dynamic_frame.from_options`.

El particionamiento es una técnica de optimización que reduce el escaneo de datos. Para obtener más información sobre el proceso de identificación de cuándo esta técnica es adecuada, consulte Reducir la cantidad de datos escaneados en la guía de Prácticas recomendadas para el ajuste del rendimiento de AWS Glue para trabajos de Apache Spark en las Recomendaciones de AWS.

Filtrado previo con predicados de inserción

En muchos casos, puede utilizar un predicado de inserción para filtrar por particiones sin tener que enumerar y leer todos los archivos del conjunto de datos. En lugar de leer todo el conjunto de datos y, a continuación, realizar el filtrado en un objeto DynamicFrame, puede aplicar el filtro directamente en los metadatos de partición en el Data Catalog. A continuación, enumere y lea solo lo que necesita realmente en un objeto DynamicFrame.

Por ejemplo, en Python podría escribir lo siguiente.

```
glue_context.create_dynamic_frame.from_catalog(
    database = "my_S3_data_set",
    table_name = "catalog_data_table",
    push_down_predicate = my_partition_predicate)
```

De este modo se crea un objeto DynamicFrame que solo carga las particiones en el Data Catalog que cumplan la expresión de predicado. En función de lo pequeño que sea un subconjunto de los datos que esté cargando, se puede ahorrar mucho tiempo de procesamiento.

La expresión de predicado puede ser cualquier expresión booleana que admita Spark SQL. Funciona todo lo que podría incluir en una cláusula `WHERE` de una consulta SQL Spark. Por ejemplo, la expresión de predicado `pushDownPredicate = "(year=='2017' and month=='04')"` solo carga las particiones en el Data Catalog que tienen tanto `year` igual que 2017 como `month` igual que 04. Para obtener más información, consulte la documentación de Apache Spark SQL y, en concreto, la referencia de funciones SQL de Scala.

Filtrado del lado del servidor mediante predicados de partición de catálogo

La opción `push_down_predicate` se aplica después de crear el listado de todas las particiones del catálogo y antes de crear el listado de los archivos de Amazon S3 para esas particiones. Si tiene muchas particiones para una tabla, el listado de particiones del catálogo puede seguir incurriendo en sobrecarga de tiempo adicional. Para abordar esta sobrecarga, puede usar la poda de particiones del lado del servidor con la opción `catalogPartitionPredicate` que utiliza índices de partición en AWS Glue Data Catalog. Esto hace que el filtrado de particiones sea mucho más rápido cuando tiene millones de particiones en una tabla. Puede usar ambos `push_down_predicate` y `catalogPartitionPredicate` en `additional_options` en forma conjunta, si su `catalogPartitionPredicate` requiere sintaxis de predicado que aún no se soporta con los índices de partición del catálogo.

Python:

```
dynamic_frame = glueContext.create_dynamic_frame.from_catalog(
    database=dbname,
    table_name=tablename,
    transformation_ctx="datasource0",
    push_down_predicate="day>=10 and customer_id like '10%'",
    additional_options={"catalogPartitionPredicate":"year='2021' and month='06'"}
)
```

Scala:

```
val dynamicFrame = glueContext.getCatalogSource(
    database = dbname,
    tableName = tablename,
    transformationContext = "datasource0",
    pushDownPredicate="day>=10 and customer_id like '10%'",
    additionalOptions = JsonOptions("""{
        "catalogPartitionPredicate": "year='2021' and month='06'"}""")
    ).getDynamicFrame()
```

> **Note**
>
> `push_down_predicate` y `catalogPartitionPredicate` utilizan sintaxis diferentes. El primero utiliza la sintaxis estándar de Spark SQL y el segundo utiliza el analizador JSQL.

Escritura de particiones

De forma predeterminada, un objeto DynamicFrame no se particiona cuando se escribe. Todos los archivos de salida se escriben en el nivel superior de la ruta de salida especificada. Hasta hace poco tiempo, la única forma de escribir un objeto DynamicFrame en particiones era convertirlo en un objeto DataFrame de Spark SQL antes de la escritura.

Sin embargo, los objetos DynamicFrame ahora admiten la partición nativa mediante una secuencia de claves, utilizando la opción `partitionKeys` al crear un receptor. Por ejemplo, el siguiente código Python escribe un conjunto de datos en Amazon S3 en formato Parquet, en directorios particionados por el tipo de campo. Desde ahí puede procesar estas particiones con otros sistemas, como Amazon Athena.

```
glue_context.write_dynamic_frame.from_options(
    frame = projectedEvents,
    connection_type = "s3",
    connection_options = {"path": "$outpath", "partitionKeys": ["type"]},
    format = "parquet")
```

Lectura de archivos de entrada en grupos más grandes

Puede configurar propiedades de las tablas para habilitar que un trabajo de ETL de AWS Glue agrupe archivos cuando se leen desde un almacén de datos de Amazon S3. Estas propiedades

permiten que cada tarea de ETL lea un grupo de archivos de entrada en una sola partición en memoria. Esto es especialmente útil cuando hay una gran cantidad de archivos pequeños en el almacén de datos de Amazon S3. Cuando configura determinadas propiedades, le indica a AWS Glue que agrupe los archivos en una partición de datos de Amazon S3 y configura el tamaño de los grupos que se leerán. También puede establecer estas opciones al leer desde un almacén de datos de Amazon S3 con el método `create_dynamic_frame.from_options`.

Para habilitar la agrupación de archivos para una tabla, establezca los pares clave-valor y en el campo de parámetros de la estructura de tabla. Utilice la notación JSON para establecer un valor del campo de parámetros de la tabla. Para obtener más información acerca de la edición de las propiedades de una tabla, consulte Visualización y edición de los detalles de la tabla.

Puede utilizar este método para habilitar la agrupación de tablas en Data Catalog con almacenes de datos de Amazon S3.

groupFiles

Establezca groupFiles (agrupar archivos) en `inPartition` para habilitar la agrupación de archivos en una partición de datos de Amazon S3. AWS Glue habilita automáticamente la agrupación si hay más de 50 000 archivos de entrada, como en el ejemplo siguiente.

```
'groupFiles': 'inPartition'
```

groupSize

Establezca groupSize en tamaño de destino de los grupos en bytes. La propiedad groupSize es opcional; si no se proporciona, AWS Glue calcula un tamaño para utilizar todos los núcleos de CPU del clúster a la vez que reduce el número total de tareas de ETL y particiones en memoria.

Por ejemplo, la entrada siguiente establece el tamaño del grupo en 1 MB.

```
'groupSize': '1048576'
```

Tenga en cuenta que `groupsize` se debe establecer con el resultado de un cálculo. Por ejemplo, 1024 * 1024 = 1048576.

recurse

Configure recurse en `True` para que lea de forma recursiva los archivos en todos los subdirectorios cuando especifique `paths` como una matriz de rutas. No es necesario configurar acción recursiva si `paths` es una matriz de claves de objeto en Amazon S3, o si el formato de entrada es parquet/orc, como en el siguiente ejemplo.

```
'recurse':True
```

Si va a leer de Amazon S3 directamente con el método `create_dynamic_frame.from_options`, agregue estas opciones de conexión. Por ejemplo, el siguiente código intenta agrupar los archivos en grupos de 1 MB.

```
df = glueContext.create_dynamic_frame.from_options("s3", {'paths': ["s3://s3path/"],
 'recurse':True, 'groupFiles': 'inPartition', 'groupSize': '1048576'}, format="json")
```

> **Note**
>
> `groupFiles` es compatible con los objetos DynamicFrames creados a partir de los siguientes formatos de datos: csv, ion, GrokLog, json y xml. Esta opción no es compatible con avro, parquet y orc.

Puntos de enlace de Amazon VPC para Amazon S3

Por razones de seguridad, muchos clientes de AWS ejecutan sus aplicaciones dentro de un entorno de Amazon Virtual Private Cloud (Amazon VPC). Con Amazon VPC, puede lanzar instancias de Amazon EC2 en una nube virtual privada que está aislada de forma lógica de otras redes, incluida la red pública de Internet. Con una Amazon VPC, puede controlar el rango de direcciones IP, las subredes, las tablas de enrutamiento, las gateways de red y los ajustes de seguridad.

> **Note**
>
> Si ha creado su cuenta de AWS después del 04/12/2013, ya dispone de una VPC predeterminada en cada región de AWS. Puede comenzar a utilizar inmediatamente su VPC predeterminada sin necesidad de realizar ningún ajuste adicional.
> Para obtener más información, consulte Su VPC y subredes predeterminadas en la Guía del usuario de Amazon VPC.

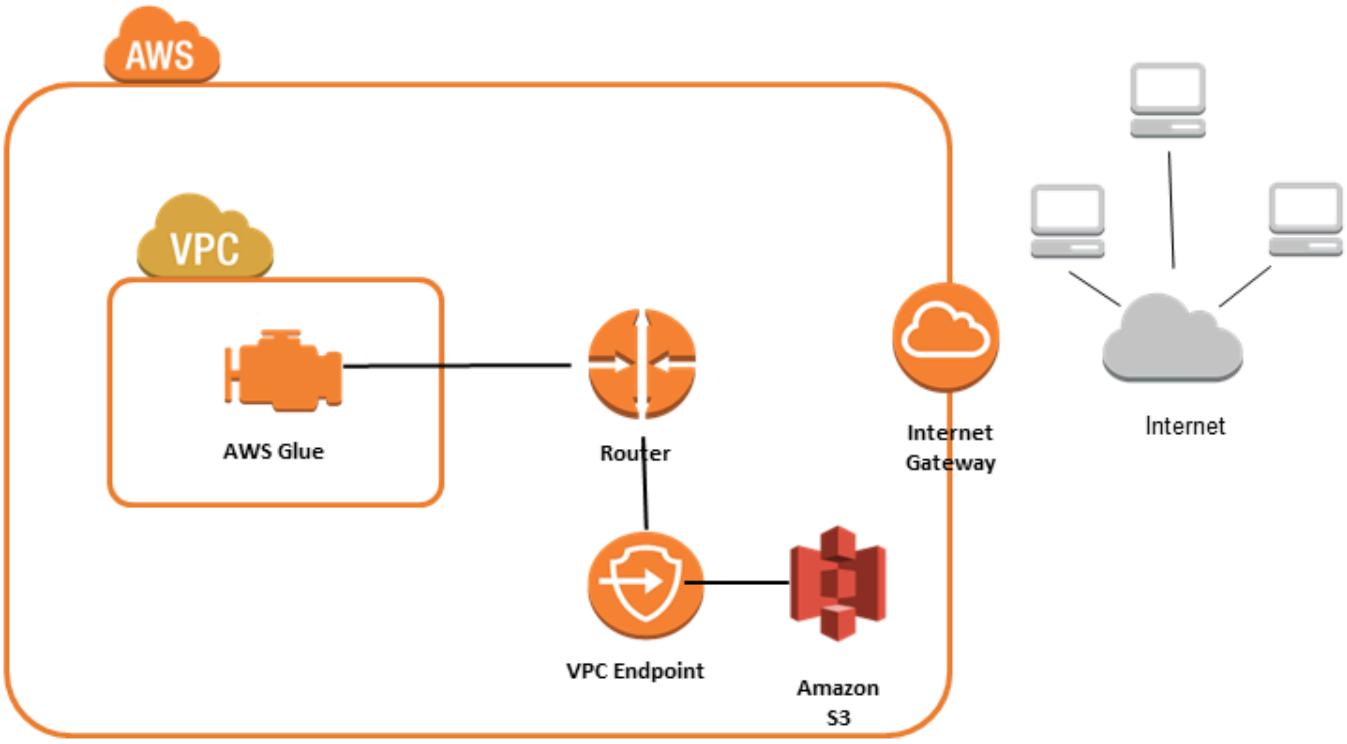
A muchos clientes les preocupa con razón la privacidad y la seguridad en el envío y recepción de datos a través de la red pública de Internet. Los clientes pueden abordar estas inquietudes con el uso de una red privada virtual (VPN) para dirigir todo el tráfico de la red de Amazon S3 a través de su propia infraestructura de red corporativa. Sin embargo, este enfoque puede conllevar problemas de ancho de banda y disponibilidad.

Estos problemas se pueden solucionar con puntos de enlace de la VPC para Amazon S3. Un punto de enlace de la VPC para Amazon S3 permite a AWS Glue utilizar direcciones IP privadas para obtener acceso a Amazon S3 sin exponerse en la red pública de Internet. AWS Glue no necesita direcciones IP públicas y el usuario no necesita una gateway de Internet, un dispositivo NAT o una gateway privada virtual en su VPC. Para controlar el acceso a Amazon S3 se utilizan políticas de punto de enlace. El tráfico entre su VPC y el servicio AWS no sale de la red de Amazon.

Cuando crea un punto de enlace de la VPC para Amazon S3, todas las solicitudes a un punto de enlace de Amazon S3 dentro de la región (por ejemplo, s3.us-west-2.amazonaws.com) se dirigen a un punto de enlace de Amazon S3 privado dentro de la red de Amazon. No necesita modificar las aplicaciones que se ejecutan en instancias de Amazon EC2 de su VPC: el nombre del punto de enlace sigue siendo el mismo, pero la ruta a Amazon S3 permanece por completo dentro de la red de Amazon y no tiene acceso a la red pública de Internet.

Para obtener más información, consulte Puntos de enlace de la VPC en la Guía del usuario de Amazon VPC.

En el siguiente diagrama se muestra cómo AWS Glue utiliza un punto de enlace de la VPC para obtener acceso a Amazon S3.



Para configurar el acceso a Amazon S3

1. Inicie sesión en la AWS Management Console y abra la consola de Amazon VPC en https://console.aws.amazon.com/vpc/.
2. En el panel de navegación izquierdo, seleccione Puntos de conexión.
3. Elija Create Endpoint (Crear punto de enlace) y siga los pasos para crear un punto de enlace de Amazon S3 en la VPC de tipo gateway.

Conexiones a Amazon DocumentDB

Puede utilizar AWS Glue para Spark para leer y escribir en tablas en bases de datos de Amazon DocumentDB. Puede conectar Amazon DocumentDB con las credenciales almacenadas en AWS Secrets Manager a través de una conexión de AWS Glue.

Para obtener más información sobre Amazon DocumentDB, consulte la documentación de Amazon DocumentDB.

> **Note**
>
> Los clústeres elásticos de Amazon DocumentDB no se admiten actualmente cuando se utiliza el conector Glue AWS. Para obtener más información sobre los clústeres elásticos, consulte Uso de clústeres elásticos de Amazon DocumentDB.

Lectura y escritura en las colecciones de Amazon DocumentDB

> **Note**
>
> Cuando crea un trabajo de ETL que se conecta a Amazon DocumentDB, para la propiedad del trabajo `Connections`, debe designar un objeto de conexión que especifique la nube privada virtual (VPC) en la que se ejecuta Amazon DocumentDB. Para el objeto de conexión, el tipo de conexión debe ser `JDBC` y el `JDBC URL` debe ser `mongo://<DocumentDB_host>:27017`.

> **Note**
>
> Estos ejemplos de código fueron desarrollados para AWS Glue 3.0. Para migrar a AWS Glue 4.0, consulte the section called “MongoDB”. El parámetro `uri` ha cambiado.

> **Note**
>
> Cuando se utiliza Amazon DocumentDB, `retryWrites` debe configurarse en falso en determinadas situaciones, como cuando el documento escrito lo especifica `_id`. Para obtener más información, consulte Diferencias funcionales con MongoDB en la documentación de Amazon DocumentDB.

El siguiente script de Python muestra el uso de tipos de conexión y opciones de conexión para leer y escribir en Amazon DocumentDB.

```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext, SparkConf
from awsglue.context import GlueContext
from awsglue.job import Job
import time

## @params: [JOB_NAME]
args = getResolvedOptions(sys.argv, ['JOB_NAME'])

sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session

job = Job(glueContext)
job.init(args['JOB_NAME'], args)

output_path = "s3://some_bucket/output/" + str(time.time()) + "/"
documentdb_uri = "mongodb://<mongo-instanced-ip-address>:27017"
documentdb_write_uri = "mongodb://<mongo-instanced-ip-address>:27017"

read_docdb_options = {
    "uri": documentdb_uri,
    "database": "test",
    "collection": "coll",
    "username": "username",
    "password": "1234567890",
    "ssl": "true",
    "ssl.domain_match": "false",
    "partitioner": "MongoSamplePartitioner",
    "partitionerOptions.partitionSizeMB": "10",
    "partitionerOptions.partitionKey": "_id"
}

write_documentdb_options = {
    "retryWrites": "false",
    "uri": documentdb_write_uri,
    "database": "test",
    "collection": "coll",
    "username": "username",
    "password": "pwd"
}
```

```
# Get DynamicFrame from  DocumentDB
dynamic_frame2 =
 glueContext.create_dynamic_frame.from_options(connection_type="documentdb",

 connection_options=read_docdb_options)

# Write DynamicFrame to MongoDB and DocumentDB
glueContext.write_dynamic_frame.from_options(dynamic_frame2,
 connection_type="documentdb",

 connection_options=write_documentdb_options)

job.commit()
```

El siguiente script de Scala muestra el uso de tipos de conexión y opciones de conexión para leer y escribir en Amazon DocumentDB.

```
import com.amazonaws.services.glue.GlueContext
import com.amazonaws.services.glue.MappingSpec
import com.amazonaws.services.glue.errors.CallSite
import com.amazonaws.services.glue.util.GlueArgParser
import com.amazonaws.services.glue.util.Job
import com.amazonaws.services.glue.util.JsonOptions
import com.amazonaws.services.glue.DynamicFrame
import org.apache.spark.SparkContext
import scala.collection.JavaConverters._

object GlueApp {
  val DOC_URI: String = "mongodb://<mongo-instanced-ip-address>:27017"
  val DOC_WRITE_URI: String = "mongodb://<mongo-instanced-ip-address>:27017"
  lazy val documentDBJsonOption = jsonOptions(DOC_URI)
  lazy val writeDocumentDBJsonOption = jsonOptions(DOC_WRITE_URI)
  def main(sysArgs: Array[String]): Unit = {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)
    val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB_NAME").toArray)
    Job.init(args("JOB_NAME"), glueContext, args.asJava)

    // Get DynamicFrame from DocumentDB
    val resultFrame2: DynamicFrame = glueContext.getSource("documentdb",
 documentDBJsonOption).getDynamicFrame()
```

```
    // Write DynamicFrame to DocumentDB
    glueContext.getSink("documentdb", writeJsonOption).writeDynamicFrame(resultFrame2)

    Job.commit()
  }

  private def jsonOptions(uri: String): JsonOptions = {
    new JsonOptions(
      s"""{"uri": "${uri}",
         |"database":"test",
         |"collection":"coll",
         |"username": "username",
         |"password": "pwd",
         |"ssl":"true",
         |"ssl.domain_match":"false",
         |"partitioner": "MongoSamplePartitioner",
         |"partitionerOptions.partitionSizeMB": "10",
         |"partitionerOptions.partitionKey": "_id"}""".stripMargin)
  }
}
```

Referencia de opción de conexión de Amazon DocumentDB

Designa una conexión a Amazon DocumentDB (con compatibilidad con MongoDB).

Las opciones de conexión son distintas entre una conexión de origen y una conexión de receptor.

"connectionType": "Documentdb" como origen

Utilice las siguientes opciones de conexión con `"connectionType": "documentdb"` como origen:

- `"uri"`: (obligatorio) el host de Amazon DocumentDB del que se va a leer, con formato `mongodb://<host>:<port>`.
- `"database"`: (obligatorio) la base de datos de Amazon DocumentDB de la que se va a leer.
- `"collection"`: (obligatorio) la recopilación de Amazon DocumentDB de la que se va a leer.
- `"username"`: (obligatorio) nombre de usuario de Amazon DocumentDB.
- `"password"`: (obligatorio) la contraseña de Amazon DocumentDB.
- `"ssl"`: (obligatorio si usa SSL) si su conexión usa SSL, debe incluir esta opción con el valor `"true"`.

- `"ssl.domain_match"`: (obligatorio si usa SSL) si su conexión usa SSL, debe incluir esta opción con el valor `"false"`.
- `"batchSize"`: (opcional): el número de documentos que se deben devolver por lote, que se utilizan dentro del cursor de lotes internos.
- `"partitioner"`: (opcional): el nombre de la clase del particionador para leer los datos de entrada de Amazon DocumentDB. El conector proporciona los siguientes particionadores:
  - `MongoDefaultPartitioner` (predeterminado) (No compatible con AWS Glue 4.0)
  - `MongoSamplePartitioner` (No es compatible con AWS Glue 4.0)
  - `MongoShardedPartitioner`
  - `MongoSplitVectorPartitioner`
  - `MongoPaginateByCountPartitioner`
  - `MongoPaginateBySizePartitioner` (No es compatible con AWS Glue 4.0)
- `"partitionerOptions"` (opcional): opciones para el particionador designado. Se admiten las siguientes opciones para cada particionador:
  - `MongoSamplePartitioner`: `partitionKey`, `partitionSizeMB`, `samplesPerPartition`
  - `MongoShardedPartitioner`: `shardkey`
  - `MongoSplitVectorPartitioner`: `partitionKey`, partitionSizeMB
  - `MongoPaginateByCountPartitioner`: `partitionKey`, `numberOfPartitions`
  - `MongoPaginateBySizePartitioner`: `partitionKey`, partitionSizeMB

  Para obtener más información acerca de estas opciones, consulte Partitioner Configuration en la documentación de MongoDB.

"connectionType": "Documentdb" como receptor

Utilice las siguientes opciones de conexión con `"connectionType": "documentdb"` como receptor:

- `"uri"`: (obligatorio) el host de Amazon DocumentDB al que se va a escribir, con formato `mongodb://<host>:<port>`.
- `"database"`: (obligatorio) la base de datos de Amazon DocumentDB a la que se va a escribir.
- `"collection"`: (obligatorio) la recopilación de Amazon DocumentDB a la que se va a escribir.
- `"username"`: (obligatorio) nombre de usuario de Amazon DocumentDB.

- `"password"`: (obligatorio) la contraseña de Amazon DocumentDB.
- `"extendedBsonTypes"`: (opcional) si se establece en `true`, permite los tipos de BSON extendidos al escribir datos en Amazon DocumentDB. El valor predeterminado es `true`.
- `"replaceDocument"`: (opcional) si es `true`, reemplaza todo el documento al guardar conjuntos de datos que contienen un campo `_id`. Si es `false`, solo se actualizan los campos del documento que coinciden con los campos del conjunto de datos. El valor predeterminado es `true`.
- `"maxBatchSize"`: (opcional): el tamaño máximo del lote para operaciones en bloque al guardar datos. El valor predeterminado es 512.
- `"retryWrites"`: (Opcional): reintenta automáticamente determinadas operaciones de escritura una sola vez si AWS Glue detecta un error de red.

Conexiones de OpenSearch Service

Puede usar AWS Glue for Spark para leer y escribir en tablas de OpenSearch Service en AWS Glue 4.0 y versiones posteriores. Puede definir qué leer del servicio OpenSearch con una consulta de OpenSearch. Se conecta a OpenSearch Service mediante credenciales de autenticación básica HTTP almacenadas AWS Secrets Manager a través de una conexión de AWS Glue. Esta característica no es compatible con OpenSearch Service sin servidor.

Para obtener más información acerca de Amazon OpenSearch Service, consulte la documentación de Amazon OpenSearch Service.

Configurar conexiones de OpenSearch Service

Para conectarse a OpenSearch Service desde AWS Glue, deberá crear y almacenar sus credenciales de OpenSearch Service en un AWS Secrets Manager secreto y, a continuación, asociar ese secreto a una conexión OpenSearch Service AWS Glue.

Requisitos previos:

- Identifique el punto de conexión del dominio, el *`AOSEndpoint`* y el puerto, *`AOSport`* desde el que desea leer o cree el recurso siguiendo las instrucciones de la documentación de Amazon OpenSearch Service. Para obtener más información sobre la creación de un dominio, consulte Crear y administrar dominios de Amazon OpenSearch Service en la documentación de Amazon OpenSearch Service.

  Un punto de conexión de dominio de Amazon OpenSearch Service tendrá el siguiente formulario predeterminado: https://

search-*domainName*-*unstructuredIdContent*.*region*.es.amazonaws.com. Para obtener más información sobre cómo identificar su punto de conexión de dominio, consulte Crear y administrar dominios de Amazon OpenSearch Service en la documentación de Amazon OpenSearch Service.

Identifique o genere credenciales de autenticación básica HTTP, *aosUser* y *aosPassword* para su dominio.

Para configurar una conexión a OpenSearch Service:

1. En AWS Secrets Manager, cree un secreto con sus credenciales de OpenSearch Service. Para crear un secreto en Secrets Manager, siga el tutorial disponible en Crear un secreto AWS Secrets Manager en la documentación AWS Secrets Manager. Después de crear el secreto, conserve el nombre secreto, *secretName*, para el siguiente paso.

   - Al seleccionar pares clave/valor, genere un par para la clave `opensearch.net.http.auth.user` con el valor *aosUser*.
   - Al seleccionar pares clave/valor, genere un par para la clave `opensearch.net.http.auth.pass` con el valor *aosPassword*.

2. En la consola de AWS Glue, genere una conexión mediante los pasos que se indican en the section called “Adición de una conexión de AWS Glue”. Tras crear la conexión, conserve el nombre de la conexión, *connectionName*, para el uso futuro en AWS Glue.

   - Al seleccionar un tipo de conexión, seleccione OpenSearch Service.
   - Al seleccionar un punto de conexión de dominio, proporcione *aosEndpoint*.
   - Al seleccionar un puerto, proporcione *aosPort*.
   - Al seleccionar un secreto AWS, proporcione un *secretName*.

Tras crear una conexión AWS Glue OpenSearch Service, deberá realizar los siguientes pasos antes de ejecutar su trabajo de AWS Glue:

- Otorgue al rol de IAM asociado al permiso de su trabajo de AWS Glue para leer *secretName*.
- En la configuración del trabajo de Glue AWS, proporcione *connectionName* como una conexión de red adicional.

Lectura de los índices de OpenSearch Service

Requisitos previos:

- Un índice de OpenSearch Service del que quiera leer, *aosIndex*.
- Una conexión de AWS Glue OpenSearch Service configurada para proporcionar información de autenticación y ubicación de red. Para obtenerla, complete los pasos del procedimiento anterior, para configurar una conexión al servicio OpenSearch. Necesitará el nombre de la conexión de AWS Glue, *connectionName*.

En este ejemplo se lee un índice de Amazon OpenSearch Service. Deberá proporcionar el parámetro `pushdown`.

Por ejemplo:

```
opensearch_read = glueContext.create_dynamic_frame.from_options(
    connection_type="opensearch",
    connection_options={
        "connectionName": "connectionName",
        "opensearch.resource": "aosIndex",
        "pushdown": "true",
    }
)
```

También puede proporcionar una cadena de consultas para filtrar los resultados devueltos a su DynamicFrame. Deberá configurar `opensearch.query`.

`opensearch.query` puede tomar una cadena de parámetros de consulta de URL *queryString* o un objeto DSL JSON de consulta *queryObject*. Para obtener más información sobre la consulta DSL, consulte Query DSL en la documentación de OpenSearch. Para proporcionar una cadena de parámetros de consulta de URL, anteponga `?q=` a la consulta, tal y como lo haría en una URL completa. Para proporcionar un objeto DSL de consulta, coloque la cadena escape del objeto JSON antes de proporcionarlo.

Por ejemplo:

```
            queryObject = "{ "query": { "multi_match": { "query": "Sample", "fields":
 [ "sample" ] } } }"
```

```
            queryString = "?q=queryString"

            opensearch_read_query = glueContext.create_dynamic_frame.from_options(
    connection_type="opensearch",
    connection_options={
        "connectionName": "connectionName",
        "opensearch.resource": "aosIndex",
        "opensearch.query": queryString,
        "pushdown": "true",
    }
)
```

Para obtener más información sobre cómo crear una consulta fuera de su sintaxis específica, consulte Sintaxis de cadenas de consulta en la documentación de OpenSearch.

Al leer colecciones de OpenSearch que contengan datos de tipo matriz, debe especificar qué campos son de tipo matriz en la llamada al método mediante el parámetro `opensearch.read.field.as.array.include`.

Por ejemplo, al leer el siguiente documento, encontrará los campos de matriz `genre` y `actor`:

```
{
    "_index": "movies",
    "_id": "2",
    "_version": 1,
    "_seq_no": 0,
    "_primary_term": 1,
    "found": true,
    "_source": {
        "director": "Frankenheimer, John",
        "genre": [
            "Drama",
            "Mystery",
            "Thriller",
            "Crime"
        ],
        "year": 1962,
        "actor": [
            "Lansbury, Angela",
            "Sinatra, Frank",
            "Leigh, Janet",
            "Harvey, Laurence",
            "Silva, Henry",
```

```
            "Frees, Paul",
            "Gregory, James",
            "Bissell, Whit",
            "McGiver, John",
            "Parrish, Leslie",
            "Edwards, James",
            "Flowers, Bess",
            "Dhiegh, Khigh",
            "Payne, Julie",
            "Kleeb, Helen",
            "Gray, Joe",
            "Nalder, Reggie",
            "Stevens, Bert",
            "Masters, Michael",
            "Lowell, Tom"
        ],
        "title": "The Manchurian Candidate"
    }
}
```

En este caso, debería incluir esos nombres de campo en su llamada al método. Por ejemplo:

```
"opensearch.read.field.as.array.include": "genre,actor"
```

Si el campo de matriz está anidado dentro de la estructura del documento, consúltelo mediante la notación de puntos: `"genre,actor,foo.bar.baz"`. Esto especificaría una matriz `baz` incluida en el documento fuente a través del documento incrustado `foo`, que contiene el documento `bar` incrustado.

Escribir en tablas de OpenSearch Service

En este ejemplo, se escribe información de un DynamicFrame existente, *`dynamicFrame`* en OpenSearch Service. Si el índice ya contiene información, AWS Glue agregará los datos de su DynamicFrame. Deberá proporcionar el parámetro `pushdown`.

Requisitos previos:

- Una tabla de OpenSearch Service a la que desearía escribir. Necesitará información de la identificación para la tabla. Llamemos a esto *`tableName`*.
- Una conexión de AWS Glue OpenSearch Service configurada para proporcionar información de autenticación y ubicación de red. Para obtenerla, complete los pasos del procedimiento anterior,

para configurar una conexión al servicio OpenSearch. Necesitará el nombre de la conexión de AWS Glue, *connectionName*.

Por ejemplo:

```
glueContext.write_dynamic_frame.from_options(
    frame=dynamicFrame,
    connection_type="opensearch",
    connection_options={
      "connectionName": "connectionName",
      "opensearch.resource": "aosIndex",
    },
)
```

Referencia de opciones de conexión de OpenSearch Service

- `connectionName`: obligatorio. Se utiliza para lectura/escritura. El nombre de una conexión de AWS Glue OpenSearch Service configurada para proporcionar información de autenticación y ubicación de red a su método de conexión.
- `opensearch.resource`: obligatorio. Se utiliza para lectura/escritura. Valores válidos: nombres de índice de OpenSearch. El nombre del índice con el que interactuará su método de conexión.
- `opensearch.query`: se utiliza para leer. Valores válidos: cadena JSON escapada o, si esta cadena comienza con ?, la parte de búsqueda de una URL. Una consulta de OpenSearch que filtra lo que se debe recuperar al leerlo. Para obtener más información sobre el uso de este parámetro, consulte la sección the section called “Lea de OpenSearch Service” anterior.
- `pushdown`: obligatorio si. Se usa para leer. Valores válidos: booleano. Indica a Spark que pase las consultas de lectura a OpenSearch para que la base de datos solo devuelva los documentos relevantes.
- `opensearch.read.field.as.array.include`: necesario si se leen datos de tipo matriz. Se usa para leer. Valores válidos: listas de nombres de campos separados por comas. Especifica los campos que se van a leer como matrices de los documentos de OpenSearch. Para obtener más información sobre el uso de este parámetro, consulte la sección the section called “Lea de OpenSearch Service” anterior.

Conexiones Redshift

Puede usar AWS Glue for Spark para leer y escribir en tablas de bases de datos de Amazon Redshift. Al conectarse a las bases de datos de Amazon Redshift, AWS Glue mueve los datos a través de Amazon S3 para lograr el máximo rendimiento mediante el SQL y los comandos de Amazon Redshift. `COPY` `UNLOAD` En AWS Glue 4.0 y versiones posteriores, puede utilizar la integración de Amazon Redshift para Apache Spark para leer y escribir con optimizaciones y funciones específicas de Amazon Redshift además de las disponibles al conectarse a través de versiones anteriores.

Descubra cómo AWS Glue facilita más que nunca a los usuarios de Amazon Redshift la migración a AWS Glue para la integración de datos sin servidor y la ETL.

Configuración de las conexiones de Redshift

Para utilizar los clústeres de Amazon Redshift en AWS Glue, necesitará algunos requisitos previos:

- Un directorio de Amazon S3 para utilizar como almacenamiento temporario al leer y escribir en la base de datos.
- Una Amazon VPC que permita la comunicación entre su clúster de Amazon Redshift, su trabajo de AWS Glue y su directorio de Amazon S3.
- Permisos de IAM adecuados en el trabajo de AWS Glue y el clúster de Amazon Redshift.

Configuración de roles de IAM

Configurar la función para el clúster de Amazon Redshift

Su clúster de Amazon Redshift debe poder leer y escribir en Amazon S3 para poder integrarse con los trabajos de AWS Glue. Para ello, puede asociar los roles de IAM al clúster de Amazon Redshift al que desee conectarse. Su función debe tener una política que permita leer y escribir en su directorio temporario de Amazon S3. Su función debe tener una relación de confianza que permita al `redshift.amazonaws.com`servicio hacerlo`AssumeRole`.

Para asociar un rol de IAM a Amazon Redshift

1. Requisitos previos: un bucket o directorio de Amazon S3 utilizado para el almacenamiento temporario de archivos.
2. Identifique qué permisos de Amazon S3 necesitará su clúster de Amazon Redshift. Al mover datos hacia y desde un clúster de Amazon Redshift, los trabajos de AWS Glue emiten

declaraciones COPY y UNLOAD contra Amazon Redshift. Si su trabajo modifica una tabla en Amazon Redshift, AWS Glue también emitirá instrucciones CREATE LIBRARY. Para obtener información sobre los permisos específicos de Amazon S3 necesarios para que Amazon Redshift ejecute estas instrucciones, consulte la documentación de Amazon Redshift: Amazon Redshift: Permissions to access other Resources. AWS

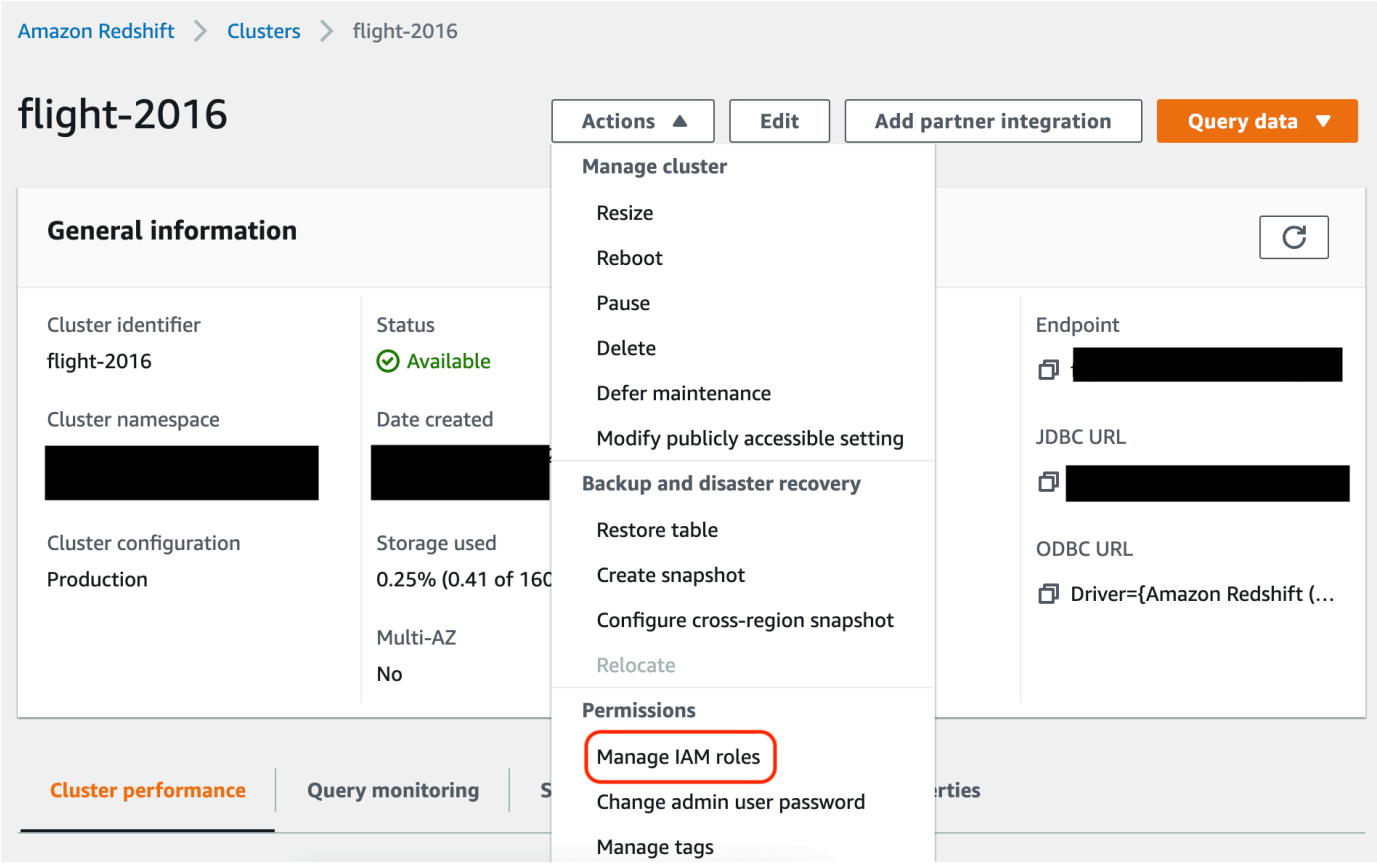
3. En la consola de IAM, cree una política de IAM con los permisos necesarios. Para obtener más información sobre cómo crear una política de IAM, consulte Creación de políticas de IAM.
4. En la consola de IAM, cree un rol y una relación de confianza que permita a Amazon Redshift asumir el rol. Siga las instrucciones de la documentación de IAM para crear un rol para un AWS servicio (consola)

   - Cuando se le pida que elija un caso de uso del AWS servicio, elija «Redshift: personalizable».
   - Cuando se pida que adjunte una política, elija la política que definió previamente.

   > **Note**
   >
   > Para obtener más información sobre la configuración de funciones para Amazon Redshift, consulte Autorizar a Amazon Redshift a acceder a otros AWS servicios en su nombre en la documentación de Amazon Redshift.

5. En la consola de Amazon Redshift, asocie la función a su clúster de Amazon Redshift. Siga las instrucciones de la documentación de Amazon Redshift.

   Seleccione la opción resaltada en la consola de Amazon Redshift para configurar este ajuste:



> **Note**
>
> De forma predeterminada, los trabajos de AWS Glue pasan las credenciales temporales de Amazon Redshift que se crean con el rol que especificó para ejecutar el trabajo. No recomendamos utilizar estas credenciales. Por motivos de seguridad, estas credenciales caducan después de 1 hora.

Configura el rol para el trabajo de AWS Glue

El trabajo de AWS Glue necesita un rol para acceder al bucket de Amazon S3. No necesita permisos de IAM para el clúster de Amazon Redshift, su acceso se controla mediante la conectividad de la VPC de Amazon y las credenciales de su base de datos.

Configuración de VPC de Amazon

Para configurar el acceso a almacenes de datos de Amazon Redshift

1. Inicie sesión en la consola de Amazon Redshift AWS Management Console y ábrala en https://console.aws.amazon.com/redshiftv2/.
2. En el panel de navegación izquierdo, elija Clusters (Clústeres).
3. Seleccione el nombre de clúster al que desee obtener acceso desde AWS Glue.
4. En la sección Cluster Properties (Propiedades del clúster), elija un grupo de seguridad en VPC security groups (Grupos de seguridad de la VPC) para permitir a AWS Glue utilizarlo. Registre el nombre del grupo de seguridad que ha elegido para futuras referencias. Al elegir el grupo de seguridad se abrirá la lista de Security Groups (Grupos de seguridad) de la consola de Amazon EC2.
5. Elija el grupo de seguridad para modificar e ir a la pestaña Inbound (Entrada).
6. Añada una regla con autorreferencia para que los componentes de AWS Glue puedan comunicarse. En concreto, añada o confirme que hay una regla con Type (Tipo) `All TCP`, Protocol (Protocolo) `TCP`, Port Range (Intervalo de puertos) con todos los puertos y el Source (Origen) con el mismo nombre de grupo de seguridad que Group ID (ID de grupo).

   La regla de entrada tiene un aspecto similar al siguiente:

Tipo	Protocolo	Intervalo de puertos	Origen
Todos los TCP	TCP	0–65535	database-security-group

   Por ejemplo:

7. Añada también una regla para el tráfico saliente. Puede abrir el tráfico saliente con destino a todos los puertos, por ejemplo:

Tipo	Protocolo	Rango de puerto	Destino
All Traffic	ALL	ALL	0.0.0.0/0

O bien, crear una regla con autorreferencia en la que Type (Tipo) sea `All TCP`, Protocol (Protocolo) sea `TCP`, Port Range (Intervalo de puertos) incluya todos los puertos y Destination (Destino) sea el mismo nombre de grupo de seguridad que Group ID (ID de grupo). En caso de que se utilice un punto de enlace de la VPC de Amazon S3, agregue también una regla HTTPS para el acceso de Amazon S3. El *`s3- prefix-list-id`* es obligatorio en la regla del grupo de seguridad para permitir el tráfico desde la VPC al punto final de la VPC de Amazon S3.

Por ejemplo:

Tipo	Protocolo	Rango de puerto	Destino
Todos los TCP	TCP	0–65535	*`security-group`*
HTTPS	TCP	443	*`s3- prefix-list-id`*

Configurar AWS Glue

Deberás crear una conexión al catálogo de datos de AWS Glue que proporcione la información de conexión de Amazon VPC.

Para configurar la conectividad de Amazon Redshift y Amazon VPC con Glue en la consola AWS

1. Cree una conexión al catálogo de datos mediante los pasos que se indican en: the section called “Adición de una conexión de AWS Glue”. Tras crear la conexión, conserve el nombre de la conexión, *`connectionName`*, para el siguiente paso.

   - Al seleccionar un tipo de conexión, seleccione Amazon Redshift.
   - Al seleccionar un clúster de Redshift, seleccione el clúster por su nombre.
   - Proporcione la información de conexión predeterminada para un usuario de Amazon Redshift en su clúster.
   - La configuración de VPC de Amazon se configurará automáticamente.

> **Note**
>
> Deberá proporcionar manualmente `PhysicalConnectionRequirements` su VPC de Amazon al crear una conexión de Amazon Redshift a través del SDK AWS .

2. En la configuración de la tarea de AWS Glue, proporciona *`ConnectionName`* como conexión de red adicional.

Ejemplo: lectura de tablas de Amazon Redshift

Puede leer los clústeres desde Amazon Redshift y los entornos de Amazon Redshift sin servidor.

Requisitos previos: una tabla de Amazon Redshift de la que quiera leer. Siga los pasos de la sección anterior, the section called “Configurar Redshift” después de lo cual debería tener el URI de Amazon S3 para un directorio temporal, *`temp-s3-dir`*, y un rol de IAM *`rs-role-name`*, (en la cuenta). *`role-account-id`*

Using the Data Catalog

Requisitos previos adicionales: una base de datos del catálogo de datos y una tabla para la tabla de Amazon Redshift de la que desee leer. Para obtener más información sobre las tablas de catálogo, consulte Detección y catalogación de datos. Tras crear una entrada para la tabla de Amazon Redshift, identificará su conexión con un *`redshift-dc-database-name`*y. *`redshift-table-name`*

Configuración: en las opciones de función, identificará la tabla del catálogo de datos con los parámetros `database` y `table_name`. Identificará su directorio temporario de Amazon S3 con `redshift_tmp_dir`. También la proporcionará *`rs-role-name`*utilizando la `aws_iam_role` clave del `additional_options` parámetro.

```
 glueContext.create_dynamic_frame.from_catalog(
    database = "redshift-dc-database-name",
    table_name = "redshift-table-name",
    redshift_tmp_dir = args["temp-s3-dir"],
    additional_options = {"aws_iam_role": "arn:aws:iam::role-account-id:role/rs-
role-name"})
```

Connecting directly

Requisitos previos adicionales: Necesitará el nombre de la tabla de Amazon Redshift (. *redshift-table-name* Necesitará la información de conexión JDBC para el clúster de Amazon Redshift que almacena esa tabla. *Deberá proporcionar la información de conexión con el host, el puerto redshift-database-name, el nombre de usuario y la contraseña.*

Puede recuperar la información de conexión de la consola de Amazon Redshift cuando trabaje con clústeres de Amazon Redshift. Si utiliza Amazon Redshift sin servidor, consulte Conexión a Amazon Redshift sin servidor en la documentación de Amazon Redshift.

Configuración: en las opciones de función, identificará los parámetros de conexión con `url`, `dbtable`, `user` y `password`. Identificará su directorio temporario de Amazon S3 con `redshift_tmp_dir`. Puede especificar su rol de IAM mediante la utilización de `aws_iam_role` cuando utiliza `from_options`. La sintaxis es similar a la de la conexión a través del catálogo de datos, pero los parámetros se colocan en el mapa `connection_options`.

Es una mala práctica codificar las contraseñas en los scripts de AWS Glue. Considera almacenar tus contraseñas AWS Secrets Manager y recuperarlas en tu script con el SDK for Python (Boto3).

```
my_conn_options = {
    "url": "jdbc:redshift://host:port/redshift-database-name",
    "dbtable": "redshift-table-name",
    "user": "username",
    "password": "password",
    "redshiftTmpDir": args["temp-s3-dir"],
    "aws_iam_role": "arn:aws:iam::account id:role/rs-role-name"
}

df = glueContext.create_dynamic_frame.from_options("redshift", my_conn_options)
```

Ejemplo: escribir en tablas de Amazon Redshift

Puede leer los clústeres desde Amazon Redshift y los entornos de Amazon Redshift sin servidor.

Requisitos previos: un clúster de Amazon Redshift y siga los pasos de la the section called "Configurar Redshift" sección anterior, tras lo cual deberá disponer del URI de Amazon S3 para un directorio temporal, *temp-s3-dir,* y un rol de IAM, (en la cuenta). *rs-role-namerole-*

account-id También necesitará un `DynamicFrame` cuyos contenido desee escribir en la base de datos.

Using the Data Catalog

Requisitos previos adicionales: una base de datos del catálogo de datos para el clúster de Amazon Redshift y una tabla a la que desee escribir. Para obtener más información sobre las tablas de catálogo, consulte Detección y catalogación de datos. Identificará su conexión y la tabla de destino con. *redshift-dc-database-nameredshift-table-name*

Configuración: en las opciones de función, identificará su base de datos del catálogo de datos con el parámetro `database` y, a continuación, proporcionará la tabla con `table_name`. Identificará su directorio temporario de Amazon S3 con `redshift_tmp_dir`. También lo proporcionará *rs-role-name*utilizando la `aws_iam_role` clave del `additional_options` parámetro.

```
 glueContext.write_dynamic_frame.from_catalog(
    frame = input dynamic frame,
    database = "redshift-dc-database-name",
    table_name = "redshift-table-name",
    redshift_tmp_dir = args["temp-s3-dir"],
    additional_options = {"aws_iam_role": "arn:aws:iam::account-id:role/rs-role-
name"})
```

Connecting through a AWS Glue connection

Puede conectarse a Amazon Redshift directamente mediante el uso del método `write_dynamic_frame.from_options`. Sin embargo, en lugar de insertar los detalles de la conexión directamente en el script, puede hacer referencia a los detalles de la conexión almacenados en una conexión del catálogo de datos con el método `from_jdbc_conf`. Puede hacerlo sin tener que rastrear ni crear tablas del catálogo de datos para la base de datos. Para obtener más información sobre las conexiones de catálogo, consulte Conexión a datos.

Requisitos previos adicionales: una conexión del catálogo de datos para su base de datos y una tabla de Amazon Redshift de la que desee leer.

Configuración: identificará su conexión al catálogo de datos con *dc-connection-name*. Identificará la base de datos y la tabla de Amazon Redshift con *redshift-table-name*y. *redshift-database-name* Deberá proporcionar la información de conexión del catálogo de datos con `catalog_connection` y la información de Amazon Redshift junto con `dbtable` y

`database`. La sintaxis es similar a la de la conexión a través del catálogo de datos, pero los parámetros se colocan en el mapa `connection_options`.

```
my_conn_options = {
    "dbtable": "redshift-table-name",
    "database": "redshift-database-name",
    "aws_iam_role": "arn:aws:iam::role-account-id:role/rs-role-name"
}

glueContext.write_dynamic_frame.from_jdbc_conf(
    frame = input dynamic frame,
    catalog_connection = "dc-connection-name",
    connection_options = my_conn_options,
    redshift_tmp_dir = args["temp-s3-dir"])
```

Referencia de opción de conexión a Amazon Redshift

Las opciones de conexión básicas que se utilizan en todas las conexiones JDBC de AWS Glue permiten configurar información similar `url` a la de todos los tipos de JDBC `user` y `password` son coherentes en todos los tipos de JDBC. Para obtener más información acerca de los parámetros de JDBC, consulte the section called “Parámetros de conexión de JDBC”.

El tipo de conexión Amazon Redshift requiere algunas opciones de conexión adicionales:

- `"redshiftTmpDir"`: (Obligatorio) La ruta de Amazon S3 donde se pueden almacenar datos temporarios al copiar desde la base de datos.
- `"aws_iam_role"`: (Opcional) ARN para un rol de IAM. La tarea de AWS Glue transferirá esta función al clúster de Amazon Redshift para conceder al clúster los permisos necesarios para completar las instrucciones de la tarea.

Opciones de conexión adicionales disponibles en AWS Glue 4.0+

También puede transferir opciones para el nuevo conector Amazon Redshift a través de las opciones de conexión de AWS Glue. Para obtener una lista completa de las opciones de conectores compatibles, consulte la sección Spark SQL parameters (Parámetros de Spark SQL) en Amazon Redshift integration for Apache Spark (Integración de Amazon Redshift para Apache Spark).

Para su comodidad, aquí reiteramos algunas opciones nuevas:

Nombre	Obligatoria	Predeterminado	Descripción
autopushdown	No	TRUE	Aplica la inserción de predicados y consultas mediante la captura y el análisis de los planes lógicos de Spark para operaciones de SQL. Las operaciones se traducen en una consulta SQL y, a continuación, se ejecutan en Amazon Redshift para mejorar el rendimiento.
autopushdown.s3_result_cache	No	FALSO	Almacena en caché la consulta SQL para descargar datos de la asignación de rutas de Amazon S3 en la memoria, de modo que no sea necesario volver a ejecutar la misma consulta en la misma sesión de Spark. Solo se admite cuando la opción `autopushdown` está habilitada.
unload_s3_format	No	PARQUET	PARQUET: descarga los resultados de la

Nombre	Obligatoria	Predeterminado	Descripción
			consulta en formato Parquet. TEXT: descarga los resultados de la consulta en formato de texto delimitado por barras.
sse_kms_key	No	N/A	La clave AWS SSE-KMS que se utilizará para el cifrado durante `UNLOAD` las operaciones en lugar de la codificación predeterminada. AWS

Nombre	Obligatoria	Predeterminado	Descripción
extracopyoptions	No	N/A	Lista de opciones adicionales que se anexarán al comando `COPY` de Amazon Redshift al cargar datos, como `TRUNCATECOLUMNS` o `MAXERROR n` (consulte otras opciones en COPY: Optional parameters [COPY: parámetros opcionales]). Tenga en cuenta que, dado que estas opciones se anexan al final del comando `COPY`, solo se pueden usar las opciones que tengan sentido al final del comando. Eso debería abarcar la mayoría de los casos de uso posibles.
csvnullstring (experimental)	No	NULL	Valor de cadena que se escribirá para los valores nulos cuando se utiliza el valor de `tempformat` CSV. Debe ser un valor que no aparezca en los datos reales.

Estos nuevos parámetros se pueden utilizar de las siguientes maneras.

Nuevas opciones para mejorar el rendimiento

El nuevo conector presenta algunas opciones nuevas de mejora del rendimiento:

- `autopushdown`: habilitada de forma predeterminada.
- `autopushdown.s3_result_cache`: deshabilitada de forma predeterminada.
- `unload_s3_format`: `PARQUET` de forma predeterminada.

Para obtener información sobre el uso de estas opciones, consulte Amazon Redshift integration for Apache Spark (Integración de Amazon Redshift para Apache Spark). Le recomendamos que no active `autopushdown.s3_result_cache` cuando tenga operaciones de lectura y escritura mixtas, ya que los resultados almacenados en caché pueden contener información obsoleta. La opción `unload_s3_format` se define como `PARQUET` de forma predeterminada para el comando `UNLOAD` a fin de mejorar el rendimiento y reducir el costo de almacenamiento. Para utilizar el comportamiento predeterminado del comando `UNLOAD`, restablezca la opción a `TEXT`.

Nueva opción de cifrado para lectura

De forma predeterminada, los datos de la carpeta temporal que AWS Glue utiliza al leer datos de la tabla de Amazon Redshift se cifran mediante el cifrado `SSE-S3`. Si quieres usar claves administradas por el cliente desde AWS Key Management Service (AWS KMS) para cifrar tus datos, puedes configurar el identificador de clave que `("sse_kms_key" # kmsKey)` corresponde a KsmKey AWS KMS, en lugar de usar la opción de configuración antigua de la versión 3.0. `("extraunloadoptions" # s"ENCRYPTED KMS_KEY_ID '$kmsKey'")` AWS Glue

```
datasource0 = glueContext.create_dynamic_frame.from_catalog(
  database = "database-name",
  table_name = "table-name",
  redshift_tmp_dir = args["TempDir"],
  additional_options = {"sse_kms_key":"<KMS_KEY_ID>"},
  transformation_ctx = "datasource0"
)
```

Compatibilidad con URL de JDBC basadas en IAM

El nuevo conector admite una URL de JDBC basada en IAM, por lo que no es necesario pasar un usuario o contraseña ni un secreto. Con una URL de JDBC basada en IAM, el conector utiliza el rol de tiempo de ejecución del trabajo para acceder al origen de datos de Amazon Redshift.

Paso 1: adjunte la siguiente política mínima requerida a su rol de tiempo de ejecución del trabajo de AWS Glue.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "redshift:GetClusterCredentials",
            "Resource": [
                "arn:aws:redshift:<region>:<account>:dbgroup:<cluster name>/*",
                "arn:aws:redshift:*:<account>:dbuser:*/*",
                "arn:aws:redshift:<region>:<account>:dbname:<cluster name>/<database
 name>"
            ]
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": "redshift:DescribeClusters",
            "Resource": "*"
        }
    ]
}
```

Paso 2: utilice la URL de JDBC basada en IAM de la siguiente manera. Especifique una nueva opción `DbUser` con el nombre de usuario de Amazon Redshift con el que se está conectando.

```
conn_options = {
     // IAM-based JDBC URL
    "url": "jdbc:redshift:iam://<cluster name>:<region>/<database name>",
    "dbtable": dbtable,
    "redshiftTmpDir": redshiftTmpDir,
    "aws_iam_role": aws_iam_role,
    "DbUser": "<Redshift User name>" // required for IAM-based JDBC URL
    }

redshift_write = glueContext.write_dynamic_frame.from_options(
    frame=dyf,
    connection_type="redshift",
    connection_options=conn_options
)
```

```
redshift_read = glueContext.create_dynamic_frame.from_options(
    connection_type="redshift",
    connection_options=conn_options
)
```

> **Note**
>
> Actualmente, un `DynamicFrame` solo admite una URL de JDBC basada en IAM con un valor de `DbUser` en el flujo de trabajo de `GlueContext.create_dynamic_frame.from_options`.

Migración de la versión 3.0 de AWS Glue a la versión 4.0

En AWS Glue 4.0, los trabajos de ETL tienen acceso a un nuevo conector Amazon Redshift Spark y a un nuevo controlador JDBC con diferentes opciones y configuraciones. El nuevo conector y controlador de Amazon Redshift están diseñados con el rendimiento como objetivo y mantienen la coherencia transaccional de los datos. Estos productos están registrados en la documentación de Amazon Redshift. Para obtener más información, consulte:

- Amazon Redshift integration for Apache Spark (Integración de Amazon Redshift para Apache Spark)
- Amazon Redshift JDBC driver, version 2.1 (Controlador JDBC versión 2.1 de Amazon Redshift)

Restricción de identificadores y nombres de tablas o columnas

El nuevo conector y el controlador de Spark para Amazon Redshift tienen un requisito más restringido para el nombre de tablas de Redshift. Si necesita más información, consulte Nombres e identificadores para definir el nombre de una tabla de Amazon Redshift. Es posible que el flujo de trabajo de marcador de trabajo no funcione con un nombre de tabla que no coincida con las reglas y con ciertos caracteres, como un espacio.

Si tiene tablas heredadas con nombres que no se ajustan a las reglas indicadas en Nombres e identificadores y tiene problemas con los marcadores (los trabajos vuelven a procesar datos antiguos de tablas de Amazon Redshift), le recomendamos que cambie el nombre de las tablas. Para obtener más información, consulte Ejemplos de ALTER TABLE.

Cambio de tempformat predeterminado en DataFrame

El conector de Spark de la versión 3.0 de AWS Glue establece de forma predeterminada el valor de `tempformat` en CSV al escribir en Amazon Redshift. Para mantener la coherencia, en la versión 3.0 de AWS Glue, el `DynamicFrame` sigue estableciendo el valor predeterminado de `tempformat` para usar `CSV`. Si ha utilizado con anterioridad las API de Dataframe de Spark directamente con el conector de Spark para Amazon Redshift, puede establecer el `tempformat` en CSV en las opciones `DataframeReader` o `Writer`. De lo contrario, `tempformat` adopta `AVRO` como valor predeterminado en el nuevo conector de Spark.

Cambio de conducta: asignación del tipo de datos REAL de Amazon Redshift al tipo de datos FLOAT de Spark en lugar de DOUBLE

En la versión 3.0 de AWS Glue, el tipo `REAL` de Amazon Redshift se convierte a un tipo `DOUBLE` de Spark. El nuevo conector de Spark para Amazon Redshift ha actualizado el comportamiento para que el tipo `REAL` de Amazon Redshift se convierta al tipo `FLOAT` de Spark y viceversa. Si tiene un caso de uso heredado en el que aún desea que el tipo `REAL` de Amazon Redshift se asigne a un tipo `DOUBLE` de Spark, puede utilizar la siguiente solución alternativa:

- En el caso de un `DynamicFrame`, asigna el tipo `Float` a un tipo `Double` con `DynamicFrame.ApplyMapping`. En el caso de un `Dataframe`, tiene que usar `cast`.

Ejemplo de código:

```
dyf_cast = dyf.apply_mapping([('a', 'long', 'a', 'long'), ('b', 'float', 'b',
 'double')])
```

Conexiones de Kafka

Designa una conexión a un clúster de Kafka o a un clúster de Amazon Managed Streaming for Apache Kafka.

Puede leer y escribir en los flujos de datos de Kafka con la información almacenada en la tabla del Catálogo de datos o al brindar la información para acceder de manera directa al flujo de datos. Puedes leer información de Kafka en una Spark DataFrame y luego convertirla en una AWS Glue DynamicFrame. Puedes escribir DynamicFrames en Kafka en formato JSON. Si accede directamente a la secuencia de datos, utilice estas opciones para proporcionar información sobre cómo acceder a la secuencia de datos.

Si utiliza `getCatalogSource` o `create_data_frame_from_catalog` para consumir los registros de un origen de streaming de Kafka, o `getCatalogSink` o

`write_dynamic_frame_from_catalog` para escribir registros en Kafka, el trabajo cuenta con la base de datos del Catálogo de datos y la información del nombre de la tabla, lo cual se puede utilizar para obtener algunos parámetros básicos para la lectura de un origen de streaming de Kafka. Si utiliza `getSource`, `getCatalogSink`, `getSourceWithFormat`, `getSinkWithFormat`, `createDataFrameFromOptions`, `create_data_frame_from_options` o `write_dynamic_frame_from_catalog`, debe especificar estos parámetros básicos con las opciones de conexión que se describen aquí.

Puede especificar las opciones de conexión para Kafka con los argumentos que se mencionan a continuación para los métodos especificados en la clase `GlueContext`.

- Scala
  - `connectionOptions`: se debe utilizar con `getSource`, `createDataFrameFromOptions` y `getSink`
  - `additionalOptions`: se debe utilizar con `getCatalogSource`, `getCatalogSink`
  - `options`: se debe utilizar con `getSourceWithFormat`, `getSinkWithFormat`
- Python
  - `connection_options`: se debe utilizar con `create_data_frame_from_options`, `write_dynamic_frame_from_options`
  - `additional_options`: se debe utilizar con `create_data_frame_from_catalog`, `write_dynamic_frame_from_catalog`
  - `options`: se debe utilizar con `getSource`, `getSink`

Para obtener notas y conocer las restricciones sobre los trabajos de ETL de transmisión, consulte the section called “Notas y restricciones de ETL de streaming”.

Configurar Kafka

No hay AWS requisitos previos para conectarse a las transmisiones de Kafka disponibles a través de Internet.

Puedes crear una conexión AWS Glue Kafka para gestionar tus credenciales de conexión. Para obtener más información, consulte the section called “Crear una conexión para un flujo de datos Kafka”. En la configuración del trabajo de AWS Glue, *`introduce ConnectionName`* como conexión de red adicional y, a continuación, en la llamada al método, introduce *`ConnectionName`* al parámetro. `connectionName`

En algunos casos, tendrá que configurar requisitos previos adicionales:

- Si utiliza Amazon Managed Streaming for Apache Kafka con autenticación de IAM, necesitará una configuración de IAM adecuada.
- Si utiliza Amazon Managed Streaming for Apache Kafka con una VPC de Amazon, necesitará una configuración de VPC de Amazon adecuada. Deberás crear una conexión AWS Glue que proporcione la información de conexión de Amazon VPC. Necesitará que la configuración de su trabajo incluya la conexión AWS Glue como conexión de red adicional.

Para obtener más información sobre los requisitos previos del trabajo de ETL de Streaming, consulte the section called “Trabajos ETL de streaming”.

Ejemplo: leer desde transmisiones de Kafka

Se utiliza junto con the section called “forEachBatch”.

Ejemplo de origen de streaming de Kafka:

```
kafka_options =
    { "connectionName": "ConfluentKafka",
      "topicName": "kafka-auth-topic",
      "startingOffsets": "earliest",
      "inferSchema": "true",
      "classification": "json"
    }
data_frame_datasource0 =
 glueContext.create_data_frame.from_options(connection_type="kafka",
 connection_options=kafka_options)
```

Ejemplo: escritura a flujos de Kafka

Ejemplos de escritura a Kafka:

Ejemplo con el método `getSink`:

```
data_frame_datasource0 =
glueContext.getSink(
 connectionType="kafka",
 connectionOptions={
  JsonOptions("""{
   "connectionName": "ConfluentKafka",
```

```
   "classification": "json",
   "topic": "kafka-auth-topic",
   "typeOfData": "kafka"}
  """)},
 transformationContext="dataframe_ApacheKafka_node1711729173428")
 .getDataFrame()
```

Ejemplo con el método `write_dynamic_frame.from_options`:

```
kafka_options =
    { "connectionName": "ConfluentKafka",
      "topicName": "kafka-auth-topic",
      "classification": "json"
    }
data_frame_datasource0 =
 glueContext.write_dynamic_frame.from_options(connection_type="kafka",
 connection_options=kafka_options)
```

Referencia de opciones de conexión de Kafka

Al leer, utilice las opciones de conexión con `"connectionType": "kafka"` a continuación:

- `"bootstrap.servers"` (obligatorio): una lista de direcciones URL Bootstrap, por ejemplo, como `b-1.vpc-test-2.o4q88o.c6.kafka.us-east-1.amazonaws.com:9094`. Esta opción debe especificarse en la llamada a la API o definirse en los metadatos de la tabla en el Data Catalog.
- `"security.protocol"` (obligatorio): el protocolo que se utiliza para la comunicación con los agentes. Los valores posibles son `"SSL"` o.`"PLAINTEXT"`
- `"topicName"` (Obligatorio) Lista separada por comas de temas a los que suscribirse. Debe especificar solo una opción de `"topicName"`, `"assign"` o `"subscribePattern"`.
- `"assign"`: (Obligatorio) Una cadena JSON que especifica el valor de `TopicPartitions` para consumir. Debe especificar solo una opción de `"topicName"`, `"assign"` o `"subscribePattern"`.

  Ejemplo: “{"temaA":[0,1],"temaB":[2,4]}”
- `"subscribePattern"`: (obligatorio) cadena de expresiones regulares de Java que identifica la lista de temas a la que desea suscribirse. Debe especificar solo una opción de `"topicName"`, `"assign"` o `"subscribePattern"`.

  Ejemplo: “tema.*”

- "classification" (Obligatorio) El formato de archivo utilizado por los datos del registro. Obligatorio a menos que se proporcione a través del catálogo de datos.
- "delimiter" (Opcional) El separador de valores que se utiliza cuando classification es CSV. El valor predeterminado es “,”.
- "startingOffsets": (opcional) posición inicial en el tema de Kafka para leer los datos. Los valores posibles son "earliest" o."latest" El valor predeterminado es "latest".
- "startingTimestamp": (Opcional, solo compatible con la versión 4.0 o posterior de AWS Glue) La marca de tiempo del registro del tema de Kafka del que se van a leer los datos. Los valores posibles son una cadena de marca de tiempo en formato UTC en el patrón yyyy-mm-ddTHH:MM:SSZ (donde Z representa un desplazamiento de zona horaria UTC con un +/-. Por ejemplo, “2023-04-04T08:00:00-04:00”).

  Nota: Solo una de las opciones «StartingOffsets» o «StartingTimestamp» puede estar presente en la lista de opciones de conexión del script de streaming de AWS Glue. Si se incluyen estas dos propiedades, se producirá un error en el trabajo.
- "endingOffsets": (opcional) el punto final cuando finaliza una consulta por lotes. Los valores posibles son "latest" o una cadena JSON que especifica una compensación final para cada TopicPartition.

  Para la cadena JSON, el formato es {"topicA":{"0":23,"1":-1},"topicB":{"0":-1}}. El valor -1 como compensación representa "latest".
- "pollTimeoutMs": (opcional) tiempo de espera en milisegundos para sondear datos de Kafka en ejecutores de trabajos de Spark. El valor predeterminado es 512.
- "numRetries": (opcional) el número de veces que se reintenta antes de no obtener las compensaciones de Kafka. El valor predeterminado es 3.
- "retryIntervalMs": (opcional) tiempo en milisegundos para esperar antes de volver a intentar obtener compensaciones Kafka. El valor predeterminado es 10.
- "maxOffsetsPerTrigger": (opcional) el límite de velocidad en el número máximo de compensaciones que se procesan por intervalo de desencadenador. El número total de compensaciones especificado se divide de forma proporcional entre topicPartitions de diferentes volúmenes. El valor predeterminado es nulo, lo que significa que el consumidor lee todos las compensaciones hasta la última compensación conocida.
- "minPartitions": (opcional) el número mínimo deseado de particiones para leer desde Kafka. El valor predeterminado es nulo, lo que significa que el número de particiones de Spark es igual al número de particiones de Kafka.

- `"includeHeaders"`: (opcional) si se deben incluir los encabezados de Kafka. Cuando la opción se establece en “verdadero”, la salida de datos contendrá una columna adicional denominada “glue_streaming_kafka_headers” con el tipo `Array[Struct(key: String, value: String)]`. El valor predeterminado es “falso”. Esta opción se encuentra disponible en la versión 3.0 o posterior de AWS Glue.
- `"schema"`: (Obligatorio cuando inferSchema se establece en false) Esquema que se va a utilizar para procesar la carga. Si la clasificación es `avro`, el esquema proporcionado debe estar en el formato de esquema Avro. Si la clasificación no es `avro`, el esquema proporcionado debe estar en el formato de esquema DDL.

  A continuación, se muestran algunos ejemplos de esquemas.

  Example in DDL schema format

  ```
  'column1' INT, 'column2' STRING , 'column3' FLOAT
  ```

  Example in Avro schema format

  ```
  {
  "type":"array",
  "items":
  {
  "type":"record",
  "name":"test",
  "fields":
  [
    {
      "name":"_id",
      "type":"string"
    },
    {
      "name":"index",
      "type":
      [
        "int",
        "string",
        "float"
      ]
    }
  ]
  }
  ```

```
}
```

- `"inferSchema"`: (opcional) El valor predeterminado es “false”. Si se establece en “true”, el esquema se detectará durante el tiempo de ejecución desde la carga dentro de `foreachbatch`.
- `"avroSchema"`: (obsoleto) Parámetro utilizado para especificar un esquema de datos Avro cuando se utiliza el formato Avro. Este parámetro se ha quedado obsoleto. Utilice el parámetro `schema`.
- `"addRecordTimestamp"`: (opcional) cuando esta opción se establece en “true”, la salida de datos contendrá una columna adicional denominada “__src_timestamp” que indica la hora en la que el tema recibió el registro correspondiente. El valor predeterminado es "false". Esta opción es compatible con la versión 4.0 o posterior de AWS Glue.
- `"emitConsumerLagMetrics"`: (Opcional) Si la opción se establece en «true», emitirá las métricas correspondientes a cada lote desde el registro más antiguo recibido por el tema hasta el momento en que llegue. AWS Glue CloudWatch El nombre de la métrica es «glue.driver.streaming». maxConsumerLagInMs». El valor predeterminado es "false". Esta opción es compatible con la versión 4.0 o posterior de AWS Glue.

Al escribir, utilice las opciones de conexión con `"connectionType": "kafka"` a continuación:

- `"connectionName"`(Obligatorio) Nombre de la conexión AWS Glue utilizada para conectarse al clúster de Kafka (similar a la fuente de Kafka).
- `"topic"` (Obligatorio) Si existe una columna de temas, su valor se utiliza como el tema al momento de la escritura de la fila en Kafka, a menos que esté establecida la opción de configuración de temas. Es decir, la opción de configuración de `topic` anula la columna de temas.
- `"partition"` (Opcional) Si se especifica un número válido de partición, esta `partition` se utilizará cuando se envíe el registro.

  Si no se especifica ninguna partición pero hay una `key`, se elegirá una partición con el hash de la clave.

  Si no hay ni una `key` ni una `partition`, se elegirá una partición según la partición sticky de esos cambios cuando al menos se produzcan bytes de batch.size en la partición.
- `"key"` (Opcional) Utilizado para la partición si la `partition` es nula.
- `"classification"` (Opcional) El formato de archivo utilizado por los datos en el registro. Solo se admiten los formatos JSON, CSV y Avro.

Con el formato Avro, podemos brindar un AvroSchema personalizado para la serialización, pero tenga en cuenta que también se tiene que brindar en el origen para la deserialización. De lo contrario, utiliza Apache de forma predeterminada AvroSchema para la serialización.

Además, puede ajustar los receptores de Kafka según sea necesario al actualizar los parámetros de configuración de productor de Kafka. Tenga en cuenta que no hay una lista de permisos en las opciones de conexión; todos los pares valores clave se conservan en el receptor como se encuentran.

Sin embargo, existe una pequeña lista de opciones de rechazos que no tendrá efecto. Para obtener más información, consulte las Configuraciones específicas de Kafka.

Conexiones de Azure Cosmos DB

Puede usar AWS Glue for Spark para leer y escribir en contenedores existentes de Azure Cosmos DB mediante la API de NoSQL en AWS Glue 4.0 y versiones posteriores. Puede definir qué leer desde Azure Cosmos DB con una consulta SQL. Se conecta a Azure Cosmos DB mediante una clave de Azure Cosmos DB almacenada en AWS Secrets Manager mediante una conexión de AWS Glue.

Para obtener más información sobre Azure Cosmos DB para NoSQL, consulte la documentación de Azure.

Configuración de las conexiones de Azure Cosmos DB

Para conectarse a Azure Cosmos DB desde AWS Glue, tendrá que crear y almacenar su clave de Azure Cosmos DB en un AWS Secrets Manager secreto y, a continuación, asociar ese secreto a una conexión AWS Glue de Azure Cosmos DB.

Requisitos previos:

- En Azure, tendrá que identificar o generar una clave de Azure Cosmos DB para que la utilice AWS Glue, `cosmosKey`. Para obtener más información, consulte Acceso seguro a los datos en Azure Cosmos DB en la documentación de Azure.

Para configurar una conexión a Azure Cosmos DB:

1. En AWS Secrets Manager, cree un secreto con su clave de datos de Azure Cosmos DB. Para crear un secreto en Secrets Manager, siga el tutorial disponible en Crear un secreto AWS

Secrets Manager en la documentación AWS Secrets Manager. Después de crear el secreto, conserve el nombre secreto, *secretName*, para el siguiente paso.

- Al seleccionar pares clave/valor, genere un par para la clave `spark.cosmos.accountKey` con el valor *cosmosKey*.

2. En la consola de AWS Glue, genere una conexión mediante los pasos que se indican en the section called “Adición de una conexión de AWS Glue”. Tras crear la conexión, conserve el nombre de la conexión, *connectionName*, para el uso futuro en AWS Glue.

   - Al seleccionar un tipo de conexión, seleccione Azure Cosmos DB.
   - Al seleccionar un secreto AWS, proporcione un *secretName*.

Tras crear una conexión a AWS Glue Azure Cosmos DB, tendrá que realizar los siguientes pasos antes de ejecutar el trabajo de AWS Glue:

- Otorgue al rol de IAM asociado al permiso de su trabajo de AWS Glue para leer *secretName*.
- En la configuración del trabajo de Glue AWS, proporcione *connectionName* como una conexión de red adicional.

Lectura de contenedores de Azure Cosmos DB para NoSQL

Requisitos previos:

- Un contenedor de Azure Cosmos DB para NoSQL del que desearía leer. Necesitará la información de identificación del contenedor.

  Un contenedor de Azure Cosmos para NoSQL se identifica por su base de datos y su contenedor. Debe proporcionar los nombres de la base de datos, *cosmosDBName*, y del contenedor, *cosmosContainerName*, al conectarse a la API de Azure Cosmos para NoSQL.
- Una conexión AWS Glue Azure Cosmos DB configurada para proporcionar información de autenticación y ubicación de red. Para obtenerla, complete los pasos del procedimiento anterior, Para configurar una conexión a Azure Cosmos DB. Necesitará el nombre de la conexión de AWS Glue, *connectionName*.

Por ejemplo:

```
azurecosmos_read = glueContext.create_dynamic_frame.from_options(
```

```
    connection_type="azurecosmos",
    connection_options={
    "connectionName": connectionName,
    "spark.cosmos.database": cosmosDBName,
    "spark.cosmos.container": cosmosContainerName,
    }
)
```

También puede proporcionar una consulta SELECT SQL para filtrar los resultados devueltos a su DynamicFrame. Deberá configurar `query`.

Por ejemplo:

```
azurecosmos_read_query = glueContext.create_dynamic_frame.from_options(
    connection_type="azurecosmos",
    connection_options={
        "connectionName": "connectionName",
        "spark.cosmos.database": cosmosDBName,
        "spark.cosmos.container": cosmosContainerName,
        "spark.cosmos.read.customQuery": "query"
    }
)
```

Escribir en un contenedor de Azure Cosmos DB para NoSQL

En este ejemplo, se escribe la información de un DynamicFrame existente, *`dynamicFrame`* en Azure Cosmos DB. Si el contenedor ya contiene información, AWS Glue añadirá los datos de su DynamicFrame. Si la información del contenedor tiene un esquema diferente al de la información que escribe, se producirán errores.

Requisitos previos:

- Una tabla de Azure Cosmos DB a la que desearía escribir. Necesitará la información de identificación del contenedor. Debe crear el contenedor antes de llamar al método de conexión.

  Un contenedor de Azure Cosmos para NoSQL se identifica por su base de datos y su contenedor. Debe proporcionar los nombres de la base de datos, *`cosmosDBName`*, y del contenedor, *`cosmosContainerName`*, al conectarse a la API de Azure Cosmos para NoSQL.
- Una conexión AWS Glue Azure Cosmos DB configurada para proporcionar información de autenticación y ubicación de red. Para obtenerla, complete los pasos del procedimiento anterior,

Para configurar una conexión a Azure Cosmos DB. Necesitará el nombre de la conexión de AWS Glue, *connectionName*.

Por ejemplo:

```
azurecosmos_write = glueContext.write_dynamic_frame.from_options(
    frame=dynamicFrame,
    connection_type="azurecosmos",
    connection_options={
    "connectionName": connectionName,
    "spark.cosmos.database": cosmosDBName,
    "spark.cosmos.container": cosmosContainerName
)
```

Referencia de opciones de conexión de Azure Cosmos DB

- `connectionName`: obligatorio. Se utiliza para lectura/escritura. El nombre de una conexión de AWS Glue Azure Cosmos DB configurada para proporcionar información de autenticación y ubicación de red a su método de conexión.
- `spark.cosmos.database`: obligatorio. Se utiliza para lectura/escritura. Valores válidos: nombres de bases de datos. Nombre de base de datos de Azure Cosmos DB para NoSQL.
- `spark.cosmos.container`: obligatorio. Se utiliza para lectura/escritura. Valores válidos: nombres de contenedores. Nombre del contenedor de Azure Cosmos DB para NoSQL.
- `spark.cosmos.read.customQuery`: se utiliza para leer. Valores válidos: consultas SQL SELECT. Consulta personalizada para seleccionar los documentos que se van a leer.

Conexiones SQL Azure

Puede usar AWS Glue for Spark para leer y escribir en tablas en instancias administradas de Azure SQL en AWS Glue 4.0 y versiones posteriores. Puede definir qué leer de Azure SQL con una consulta SQL. Se conecta a Azure SQL con las credenciales de usuario y contraseña almacenadas en AWS Secrets Manager a través de una conexión de AWS Glue.

Para obtener más información sobre Azure SQL, consulte la documentación de Azure SQL.

Configuración de las conexiones de Azure SQL

Para conectarse a Azure SQL desde AWS Glue, tendrá que crear y almacenar las credenciales de Azure SQL en un AWS Secrets Manager secreto y, a continuación, asociar ese secreto a una conexión de AWS Glue de Azure SQL.

Para configurar una conexión a Azure SQL:

1. En AWS Secrets Manager, cree un secreto con sus credenciales de Azure SQL. Para crear un secreto en Secrets Manager, siga el tutorial disponible en Crear un secreto AWS Secrets Manager en la documentación AWS Secrets Manager. Después de crear el secreto, conserve el nombre secreto, *secretName*, para el siguiente paso.

   - Al seleccionar pares clave/valor, genere un par para la clave `user` con el valor *azuresqlUsername*.
   - Al seleccionar pares clave/valor, genere un par para la clave `password` con el valor *azuresqlPassword*.

2. En la consola de AWS Glue, genere una conexión mediante los pasos que se indican en the section called “Adición de una conexión de AWS Glue”. Tras crear la conexión, conserve el nombre de la conexión, *connectionName*, para el uso futuro en AWS Glue.

   - Al seleccionar un tipo de conexión, seleccione Azure SQL.
   - Al proporcionar una URL de Azure SQL, proporcione una URL de punto de conexión de JDBC.

     La lista de URL debe tener el siguiente formato: `jdbc:sqlserver://`*databaseServerName*`:`*databasePort*`;databaseName=`*azuresqlDBnam*

     AWS Glue requiere las siguientes propiedades de URL:

     - `databaseName`: una base de datos predeterminada en Azure SQL a la que conectarse.

     Para obtener más información sobre las direcciones URL de JDBC para instancias administradas de Azure SQL, consulte la documentación de Microsoft.

   - Al seleccionar un secreto AWS, proporcione un *secretName*.

Tras crear una conexión AWS Glue Azure SQL, tendrá que realizar los siguientes pasos antes de ejecutar el trabajo de AWS Glue:

- Otorgue al rol de IAM asociado al permiso de su trabajo de AWS Glue para leer *secretName*.

- En la configuración del trabajo de Glue AWS, proporcione *connectionName* como una conexión de red adicional.

Lectura de tablas de Azure SQL

Requisitos previos:

- Una tabla de Azure SQL desde la cual le gustaría leer. Necesitará información de identificación para la tabla, *databaseName* y *tableIdentifier*.

  Una tabla de Azure SQL se identifica por su base de datos, esquema y nombre de tabla. Debe proporcionar el nombre de la base de datos y el nombre de la tabla al conectarse a Azure SQL. También debe proporcionar el esquema si no es el predeterminado, "público". La base de datos se proporciona a través de una propiedad URL en *connectionName*, y el nombre de la tabla y el esquema a través de `dbtable`.
- Una conexión AWS Glue Azure SQL configurada para proporcionar información de autenticación. Complete los pasos del procedimiento anterior, Para configurar una conexión a Azure SQL para configurar su información de autenticación. Necesitará el nombre de la conexión de AWS Glue, *connectionName*.

Por ejemplo:

```
azuresql_read_table = glueContext.create_dynamic_frame.from_options(
    connection_type="azuresql",
    connection_options={
        "connectionName": "connectionName",
        "dbtable": "tableIdentifier"
    }
)
```

También puede proporcionar una consulta SELECT SQL para filtrar los resultados devueltos a su DynamicFrame. Deberá configurar `query`.

Por ejemplo:

```
azuresql_read_query = glueContext.create_dynamic_frame.from_options(
    connection_type="azuresql",
    connection_options={
        "connectionName": "connectionName",
```

```
        "query": "query"
    }
)
```

Escribir en tablas de Azure SQL

En este ejemplo, se escribe la información de un DynamicFrame o *dynamicFrame* existente en Azure SQL. Si la tabla ya contiene información, AWS Glue añadirá los datos de su DynamicFrame.

Requisitos previos:

- Una tabla de Azure SQL en la cual le gustaría escribir. Necesitará información de identificación para la tabla, *databaseName* y *tableIdentifier*.

  Una tabla de Azure SQL se identifica por su base de datos, esquema y nombre de tabla. Debe proporcionar el nombre de la base de datos y el nombre de la tabla al conectarse a Azure SQL. También debe proporcionar el esquema si no es el predeterminado, "público". La base de datos se proporciona a través de una propiedad URL en *connectionName*, y el nombre de la tabla y el esquema a través de `dbtable`.
- Información de autenticación de Azure SQL. Complete los pasos del procedimiento anterior, Para configurar una conexión a Azure SQL para configurar su información de autenticación. Necesitará el nombre de la conexión de AWS Glue, *connectionName*.

Por ejemplo:

```
azuresql_write = glueContext.write_dynamic_frame.from_options(
    connection_type="azuresql",
    connection_options={
        "connectionName": "connectionName",
        "dbtable": "tableIdentifier"
    }
)
```

Referencia de opciones de conexión de Azure SQL

- `connectionName`: obligatorio. Se utiliza para lectura/escritura. El nombre de una conexión de AWS Glue Azure SQL configurada para proporcionar información de autenticación al método de conexión.

- `databaseName`: se utiliza para lectura/escritura. Valores válidos: nombres de bases de datos de Azure SQL. Nombre de la base de datos de Azure SQL a la que se va a conectar.
- `dbtable`: necesario para escribir, obligatorio para leer a menos que se proporcione `query`. Se utiliza para lectura/escritura. Valores válidos: nombres de tablas de Azure SQL o combinaciones de nombres de tablas y esquemas separados por puntos. Se utiliza para especificar la tabla y el esquema que identifican la tabla a la que se va a conectar. El esquema predeterminado es "público". Si la tabla está en un esquema que no es el predeterminado, proporciona esta información en la forma *`schemaName`*`.`*`tableName`*.
- `query`: se utiliza para leer. Una consulta SELECT de Transact-SQL que defina lo que se debe recuperar al leer desde Azure SQL. Para obtener más información, consulte la documentación de Microsoft.

Conexiones de BigQuery

Puede usar AWS Glue para Spark para leer y escribir en tablas de Google BigQuery en AWS Glue 4.0 y versiones posteriores. Puede leer desde BigQuery con una consulta de SQL de Google. Se conecta BigQuery con las credenciales almacenadas en AWS Secrets Manager a través de una conexión de AWS Glue.

Para obtener más información sobre Google BigQuery, consulte el sitio web de Google Cloud BigQuery.

Configuración de conexiones con BigQuery

Para conectarse a Google BigQuery desde AWS Glue, tendrá que crear y almacenar las credenciales de Google Cloud Platform en un secreto de AWS Secrets Manager y, a continuación, asociar ese secreto a una conexión de Google BigQuery para AWS Glue.

Para configurar una conexión a BigQuery, siga estos pasos:

1. En Google Cloud Platform, genere e identifique los recursos pertinentes:
   - Genere e identifique un proyecto de GCP que contenga tablas de BigQuery a las que desea conectarse.
   - Habilite la API de BigQuery. Para obtener más información, consulte Cómo usar la API de lectura de almacenamiento de BigQuery para leer datos de tablas.
2. En Google Cloud Platform, genere y exporte las credenciales de la cuenta de servicio:

Puede usar el asistente de credenciales de BigQuery para simplificar este paso: Crear credenciales.

Para crear una cuenta de servicio en GCP, siga el tutorial disponible en Crear cuentas de servicio.

- Al elegir un proyecto, seleccione el proyecto que contiene su tabla de BigQuery.
- Cuando elija los roles de IAM de GCP para la cuenta de servicio, agregue o genere una función que conceda los permisos apropiados para ejecutar los trabajos en BigQuery, como la lectura, escritura o creación de tablas de BigQuery.

Para crear credenciales para la cuenta de servicio, siga el tutorial disponible en Crear una clave de cuenta de servicio.

- Al seleccionar el tipo de clave, seleccione JSON.

Ahora debería haber descargado un archivo JSON que contiene las credenciales de la cuenta de servicio. Debería parecerse a lo que sigue:

```
{
  "type": "service_account",
  "project_id": "*****",
  "private_key_id": "*****",
  "private_key": "*****",
  "client_email": "*****",
  "client_id": "*****",
  "auth_uri": "https://accounts.google.com/o/oauth2/auth",
  "token_uri": "https://oauth2.googleapis.com/token",
  "auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs",
  "client_x509_cert_url": "*****",
  "universe_domain": "googleapis.com"
}
```

3. Codifique en base64 el archivo de credenciales descargado. En una sesión AWS CloudShell o similar, puede hacerlo desde la línea de comandos al ejecutar `cat` *`credentialsFile.json`* `| base64 -w 0`. Conserve el resultado de este comando, *`credentialString`*.
4. En AWS Secrets Manager, genere un secreto mediante las credenciales de la plataforma de Google Cloud. Para crear un secreto en Secrets Manager, siga el tutorial disponible en Crear un

secreto AWS Secrets Manager en la documentación AWS Secrets Manager. Después de crear el secreto, conserve el nombre secreto, *secretName*, para el siguiente paso.

- Al seleccionar pares clave/valor, genere un par para la clave `credentials` con el valor *credentialString*.

5. En el catálogo de datos de Glue AWS, cree una conexión mediante los pasos que se indican en the section called “Adición de una conexión de AWS Glue”. Tras crear la conexión, conserve el nombre de la conexión, *connectionName*, para el siguiente paso.
   - Cuando elija un Tipo de conexión, seleccione Google BigQuery.
   - Al seleccionar un secreto AWS, proporcione un *secretName*.
6. Otorgue al rol de IAM asociado al trabajo de AWS Glue el permiso para leer *secretName*.
7. En la configuración del trabajo de Glue AWS, proporcione *connectionName* como una conexión de red adicional.

Lectura de tablas de BigQuery

Requisitos previos:

- Una tabla de BigQuery desde la cual le gustaría leer. Necesitará los nombres de la tabla y el conjunto de datos de BigQuery, en el formulario `[dataset].[table]`. Llamemos a esto *tableName*.
- El proyecto de facturación para la tabla de BigQuery. Necesitará el nombre del proyecto, *parentProject*. Si no hay ningún proyecto principal de facturación, utilice el proyecto que contiene la tabla.
- Información de autenticación de BigQuery. Complete los pasos para administrar sus credenciales de conexión con AWS Glue para configurar su información de autenticación. Necesitará el nombre de la conexión de AWS Glue, *connectionName*.

Por ejemplo:

```
bigquery_read = glueContext.create_dynamic_frame.from_options(
    connection_type="bigquery",
    connection_options={
        "connectionName": "connectionName",
        "parentProject": "parentProject",
        "sourceType": "table",
```

```
        "table": "tableName",
    }
```

También puede proporcionar una consulta para filtrar los resultados devueltos a su DynamicFrame. Tendrá que configurar `query`, `sourceType`, `viewsEnabled` y `materializationDataset`.

Por ejemplo:

Requisitos previos adicionales:

Deberá crear o identificar un conjunto de datos de BigQuery, *`materializationDataset`*, en el que BigQuery pueda escribir vistas materializadas para las consultas.

Deberá otorgar los permisos de IAM GCP adecuados a la cuenta de servicio para crear tablas en *`materializationDataset`*.

```
glueContext.create_dynamic_frame.from_options(
            connection_type="bigquery",
            connection_options={
                "connectionName": "connectionName",
                "materializationDataset": materializationDataset,
                "parentProject": "parentProject",
                "viewsEnabled": "true",
                "sourceType": "query",
                "query": "select * from bqtest.test"
            }
        )
```

Escribir en tablas de BigQuery

En este ejemplo escribe directamente en el servicio de BigQuery. BigQuery también admite el método de escritura “indirecta”. Para obtener más información sobre la configuración de escritura indirecta, consulte the section called “Uso de escritura indirecta con Google BigQuery”.

Requisitos previos:

- Una tabla de BigQuery en la cual le gustaría escribir. Necesitará los nombres de la tabla y el conjunto de datos de BigQuery, en el formulario `[dataset].[table]`. También puede proporcionar un nombre de tabla nuevo que se creará automáticamente. Llamemos a esto *`tableName`*.

- El proyecto de facturación para la tabla de BigQuery. Necesitará el nombre del proyecto, *parentProject*. Si no hay ningún proyecto principal de facturación, utilice el proyecto que contiene la tabla.
- Información de autenticación de BigQuery. Complete los pasos para administrar sus credenciales de conexión con AWS Glue para configurar su información de autenticación. Necesitará el nombre de la conexión de AWS Glue, *connectionName*.

Por ejemplo:

```
bigquery_write = glueContext.write_dynamic_frame.from_options(
    frame=frameToWrite,
    connection_type="bigquery",
    connection_options={
        "connectionName": "connectionName",
        "parentProject": "parentProject",
        "writeMethod": "direct",
        "table": "tableName",
    }
)
```

Referencia de opciones de conexión de BigQuery

- `project`: (Predeterminado): cuenta de servicio de Google Cloud predeterminada. Se utiliza para lectura/escritura. El nombre de un proyecto de Google Cloud asociado a la tabla.
- `table` — (Obligatorio) Se utiliza para lectura/escritura. El nombre de la tabla de BigQuery en el formato `[[project:]dataset.]`.
- `dataset` (Obligatorio): cuando no se ha definido mediante la opción `table`. Se utiliza para lectura/escritura. El nombre del conjunto de datos que contiene la tabla de BigQuery.
- `parentProject`: (Predeterminado): cuenta de servicio de Google Cloud predeterminada. Se utiliza para lectura/escritura. El nombre de un proyecto de Google Cloud asociado a `project` que se utiliza para la facturación.
- `sourceType`: se utiliza para leer. Obligatorio al leer. Valores válidos: `table`, `query` informa a AWS Glue si leerá por tabla o por consulta.
- `materializationDataset`: se utiliza para leer. Valores válidos: cadenas. El nombre de un conjunto de datos de BigQuery que se utiliza para almacenar las materializaciones de las vistas.

- `viewsEnabled`: se utiliza para leer. Valor predeterminado: falso. Valores válidos: true y false. Configura si BigQuery utilizará las vistas.
- `query`: se utiliza para leer. Se utiliza cuando `viewsEnabled` es verdadero. Una consulta DQL de GoogleSQL.
- `temporaryGcsBucket`: se utiliza para escribir. Obligatorio cuando `writeMethod` se establece en el valor predeterminado (`indirect`). Nombre de un bucket de Google Cloud Storage que se utiliza para almacenar una forma intermedia de los datos mientras escribe en BigQuery.
- `writeMethod` — Predeterminado: `indirect`. Valores válidos: `direct`, `indirect`. Se utiliza para escribir. Especifica el método utilizado para escribir los datos.
  - Si se establece en `direct`, el conector escribirá con la API de escritura de BigQuery Storage.
  - Si se establece en `indirect`, el conector escribirá en Google Cloud Storage y, a continuación, lo transferirá a BigQuery mediante una operación de carga. Su cuenta de servicio de Google Cloud necesitará los permisos de GCS adecuados.

Uso de escritura indirecta con Google BigQuery

En este ejemplo, se usa la escritura indirecta, que escribe los datos en Google Cloud Storage y los copia en Google BigQuery.

Requisitos previos:

Necesitará un bucket temporal de Google Cloud Storage, llamado *`temporaryBucket`*.

El rol de IAM de GCP de la cuenta de servicio de GCP de AWS Glue necesitará los permisos de GCS adecuados para acceder a *`temporaryBucket`*.

Configuración adicional:

Para configurar la escritura indirecta con BigQuery, siga estos pasos:

1. Evalúe the section called “Cómo configurar BigQuery” y localice o vuelva a descargar el archivo JSON de credenciales de GCP. Identifique *`secretName`*, el secreto de AWS Secrets Manager de la conexión Google BigQuery para AWS Glue que se utiliza en el trabajo.
2. Cargue el archivo JSON de sus credenciales a una ubicación de Amazon S3 con la seguridad adecuada. Conserve la ruta del archivo, *`s3secretpath`*, para pasos futuros.
3. Edite el *`secretName`* y agregue la clave `spark.hadoop.google.cloud.auth.service.account.json.keyfile`. Establezca el valor en *`s3secretpath`*.

4. Otorgue permisos de IAM para AWS Glue en Amazon S3 para acceder a *s3secretpath*.

Ahora puede proporcionar la ubicación temporal de su bucket de GCS a su método de escritura. No es necesario proporcionar `writeMethod`, ya que `indirect` históricamente es el valor predeterminado.

```
bigquery_write = glueContext.write_dynamic_frame.from_options(
    frame=frameToWrite,
    connection_type="bigquery",
    connection_options={
        "connectionName": "connectionName",
        "parentProject": "parentProject",
        "temporaryGcsBucket": "temporaryBucket",
        "table": "tableName",
    }
)
```

Conexiones de JDBC

Algunos tipos de bases de datos, normalmente relacionales, admiten la conexión a través del estándar JDBC. Para obtener más información sobre JDBC, consulte la Java JDBC API documentation. AWS Glue admite de forma nativa la conexión a determinadas bases de datos a través de sus conectores JDBC; las bibliotecas JDBC se proporcionan en los trabajos de Glue Spark AWS. Al conectarse a estos tipos de bases de datos mediante las bibliotecas de GlueAWS , tiene acceso a un conjunto estándar de opciones.

Entre los valores de ConnectionType de JDBC, se incluyen los siguientes:

- `"connectionType": "sqlserver"`: designa una conexión a una base de datos Microsoft SQL Server.
- `"connectionType": "mysql"`: designa una conexión a una base de datos MySQL.
- `"connectionType": "oracle"`: designa una conexión a una base de datos Oracle.
- `"connectionType": "postgresql"`: designa una conexión a una base de datos PostgreSQL.
- `"connectionType": "redshift"`: designa una conexión a una base de datos de Amazon Redshift. Para obtener más información, consulte the section called “Conexiones Redshift”.

En la siguiente tabla, se muestran las versiones del controlador JDBC que AWS Glue admite.

Producto	Versiones del controlador JDBC para Glue 4.0	Versiones del controlador JDBC para Glue 3.0	Versiones del controlador JDBC para Glue 0.9, 1.0, 2.0
Microsoft SQL Server	9.4.0	7.x	6.x
MySQL	8.0.23	8.0.23	5.1
Oracle Database	21.7	21.1	11.2
PostgreSQL	42.3.6	42.2.18	42.1.x
MongoDB	4.7.2	4.0.0	2.0.0
Amazon Redshift *	redshift-jdbc42-2.1.0.16	redshift-jdbc41-1.2.12.1017	redshift-jdbc41-1.2.12.1017

* En el caso del tipo de conexión de Amazon Redshift, todos los demás pares nombre/valor que estén presentes en las opciones de una conexión JDBC, incluidas las opciones de formato, se pasan directamente al DataSource SparkSQL subyacente. En los trabajos de Glue with Spark AWS en Glue 4.0 AWS y versiones posteriores, el conector nativo de Glue AWS para Amazon Redshift utiliza la integración de Amazon Redshift para Apache Spark. Para obtener más información, consulte Integración de Amazon Redshift para Apache Spark. En versiones anteriores, consulte el origen de datos de Amazon Redshift para Spark.

Para configurar su VPC de Amazon para que se conecte a los almacenes de datos de Amazon RDS mediante JDBC, consulte the section called “Configuración de una VPC de Amazon para conectarse a los almacenes de datos de Amazon RDS”.

> **Note**
>
> Los trabajos de AWS Glue solo se asocian a una subred durante una ejecución. Esto puede afectar a su capacidad de conectarse a diversos orígenes de datos a través del mismo trabajo. Este comportamiento no se limita a los orígenes de JDBC.

Temas

- Referencia de opciones de conexión de JDBC

- Utilice sampleQuery
- Utilice un controlador JDBC personalizado
- Lectura desde tablas de JDBC en paralelo
- Configuración de una VPC de Amazon para conexiones JDBC a los almacenes de datos de Amazon RDS de AWS Glue

Referencia de opciones de conexión de JDBC

Si ya tiene definida una conexión JDBC de Glue AWS, puede reutilizar las propiedades de configuración definidas en ella, como url, usuario y contraseña; de este modo, no tendrá que especificarlas en el código como opciones de conexión. Esta característica solo está disponible en Glue 3.0 AWS o versiones posteriores. Para ello, utilice las siguientes propiedades de conexión:

- `"useConnectionProperties"`: configúrelo en “true” para indicar que desea utilizar la configuración desde una conexión.
- `"connectionName"`: ingrese el nombre de la conexión desde la que recuperará la configuración. La conexión debe definirse en la misma región que el trabajo.

Utilice estas opciones con las conexiones JDBC:

- `"url"`: (obligatorio) la URL de JDBC de la base de datos.
- `"dbtable"`: (Obligatorio) la tabla de la base de datos de la que se va a leer. Para almacenes de datos de JDBC que admiten esquemas dentro de una base de datos, especifique `schema.table-name`. Si no se ha proporcionado un esquema, se usa el esquema "public" predeterminado.
- `"user"`: (obligatorio) nombre de usuario que se va a utilizar para establecer la conexión.
- `"password"`: (obligatorio) contraseña que se utilizará para establecer la conexión.
- (Opcional) las siguientes opciones le permiten proporcionar un controlador JDBC personalizado. Utilice estas opciones si debe utilizar un controlador que AWS Glue no admita de forma nativa.

  Los trabajos de ETL pueden utilizar diferentes versiones de controladores JDBC para el origen y el destino de los datos, incluso aunque el origen y el destino sean el mismo producto de base de datos. Esto le permite migrar datos entre bases de datos de origen y destino con diferentes versiones. Para usar estas opciones, primero debe subir el archivo JAR del controlador JDBC a Amazon S3.

  - `"customJdbcDriverS3Path"`: ruta de Amazon S3 del controlador JDBC personalizado.
  - `"customJdbcDriverClassName"`: nombre de clase del controlador JDBC.
- `"bulkSize"`: (opcional) se utiliza para configurar inserciones paralelas para acelerar las cargas masivas en los objetivos de JDBC. Especifique un valor entero para el grado de paralelismo que se utilizará al escribir o insertar datos. Esta opción resulta útil para mejorar el rendimiento de las escrituras en bases de datos como el repositorio de usuarios de Arch (AUR).
- `"hashfield"` (Opcional) Una cadena, utilizada para especificar el nombre de una columna de la tabla JDBC que se utilizará para dividir los datos en particiones al leer tablas JDBC en paralelo. Proporcione “hashfield” O “hashexpression”. Para obtener más información, consulte the section called “Lectura desde JDBC en paralelo”.
- `"hashexpression"` (Opcional) Una cláusula de selección de SQL que devuelve un número entero. Se utiliza para dividir los datos de una tabla JDBC en particiones al leer tablas JDBC en paralelo. Proporcione “hashfield” O “hashexpression”. Para obtener más información, consulte the section called “Lectura desde JDBC en paralelo”.
- `"hashpartitions"` (Opcional) Un entero positivo. Se utiliza para especificar el número de lecturas paralelas de la tabla JDBC al leer tablas JDBC en paralelo. Valor predeterminado: 7. Para obtener más información, consulte the section called “Lectura desde JDBC en paralelo”.
- `"sampleQuery"`: (Opcional) Una instrucción de consulta SQL personalizada. Se utiliza para especificar un subconjunto de información en una tabla para recuperar una muestra del contenido de la tabla. Cuando se configura sin tener en cuenta los datos, puede ser menos eficiente que los métodos DynamicFrame y provocar tiempos de espera o errores de falta de memoria. Para obtener más información, consulte the section called “Utilice sampleQuery”.
- `"enablePartitioningForSampleQuery"`: (Opcional) Un booleano. Valor predeterminado: falso. Se utiliza para permitir la lectura de tablas JDBC en paralelo al especificar `sampleQuery`. Si se establece en true, **`sampleQuery`** debe terminar con “where” (dónde) o “and” (y) para que Glue AWS agregue condiciones de partición. Para obtener más información, consulte the section called “Utilice sampleQuery”.
- `"sampleSize"`: (Opcional) Un entero positivo. Limita el número de filas que devuelve la consulta de ejemplo. Funciona solo cuando `enablePartitioningForSampleQuery` es true. Si la partición no está habilitada, en su lugar debe agregar `"limit x"` en el `sampleQuery` para limitar el tamaño. Para obtener más información, consulte the section called “Utilice sampleQuery”.

Utilice sampleQuery

En esta sección, se explica cómo utilizar `sampleQuery`, `sampleSize` y `enablePartitioningForSampleQuery`.

`sampleQuery` puede ser una forma eficaz de muestrear algunas filas de su conjunto de datos. De forma predeterminada, la consulta la ejecuta un ejecutor único. Cuando se configura sin tener en cuenta los datos, puede ser menos eficiente que los métodos DynamicFrame y provocar tiempos de espera o errores de falta de memoria. Por lo general, solo es necesario ejecutar SQL en la base de datos subyacente como parte de la canalización de ETL por motivos de rendimiento. Si intenta obtener una vista previa de algunas filas de su conjunto de datos, considere utilizar the section called "show". Si intenta transformar su conjunto de datos mediante SQL, considere utilizar the section called "toDF" para definir una transformación de SparkSQL contra sus datos en forma de DataFrame.

Si bien su consulta puede manipular una variedad de tablas, `dbtable` sigue siendo obligatoria.

Uso de sampleQuery para recuperar una muestra de la tabla

Cuando se utiliza el comportamiento predeterminado de sampleQuery para recuperar una muestra de los datos, Glue AWS no espera un rendimiento sustancial, por lo que ejecuta la consulta en un único ejecutor. Para limitar los datos que proporciona y no causar problemas de rendimiento, sugerimos que proporcione una cláusula `LIMIT` a SQL.

Example Uso de sampleQuery sin particionar

En el siguiente ejemplo de código, se muestra cómo utilizar `sampleQuery` sin particionar.

```
//A full sql query statement.
val query = "select name from $tableName where age > 0 limit 1"
val connectionOptions = JsonOptions(Map(
    "url" -> url,
    "dbtable" -> tableName,
    "user" -> user,
    "password" -> password,
    "sampleQuery" -> query ))
val dyf = glueContext.getSource("mysql", connectionOptions)
          .getDynamicFrame()
```

Uso de sampleQuery contra conjuntos de datos más grandes

Si está leyendo un conjunto de datos grande, es posible que tenga que habilitar las particiones de JDBC para consultar una tabla en paralelo. Para obtener más información, consulte the section

called “Lectura desde JDBC en paralelo”. Para utilizar `sampleQuery` con la partición JDBC,configure `enablePartitioningForSampleQuery` como true. La activación de esta característica requiere que realice algunos cambios en su `sampleQuery`.

Cuando utilice la partición JDBC con `sampleQuery`, la consulta debe terminar con “where” (dónde) o “and” (y) para que Glue AWS agregue condiciones de partición.

Si desea limitar los resultados de su sampleQuery al leer tablas JDBC en paralelo, defina el parámetro `"sampleSize"` en lugar de especificar una cláusula `LIMIT`.

Example Uso de sampleQuery con particiones JDBC

En el siguiente ejemplo de código, se muestra cómo utilizar `sampleQuery` con partición JDBC.

```
//note that the query should end with "where" or "and" if use with JDBC partitioning.
val query = "select name from $tableName where age > 0 and"

//Enable JDBC partitioning by setting hashfield.
//to use sampleQuery with partitioning, set enablePartitioningForSampleQuery.
//use sampleSize to limit the size of returned data.
val connectionOptions = JsonOptions(Map(
    "url" -> url,
    "dbtable" -> tableName,
    "user" -> user,
    "password" -> password,
    "hashfield" -> primaryKey,
    "sampleQuery" -> query,
    "enablePartitioningForSampleQuery" -> true,
    "sampleSize" -> "1" ))
val dyf = glueContext.getSource("mysql", connectionOptions)
          .getDynamicFrame()
```

Notas y restricciones:

Las consultas de muestra no se pueden utilizar junto con los marcadores de trabajo. Se ignorará el estado del marcador cuando se brinden configuraciones para ambos.

Utilice un controlador JDBC personalizado

En los ejemplos de código siguientes, se muestra cómo se utilizan controladores JDBC personalizados para leer bases de datos JDBC y escribir en ellas. En ellos, se muestra cómo se lee una versión de un producto de base de datos y se escribe en una versión posterior del mismo producto.

Python

```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext, SparkConf
from awsglue.context import GlueContext
from awsglue.job import Job
import time
from pyspark.sql.types import StructType, StructField, IntegerType, StringType

sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session

# Construct JDBC connection options
connection_mysql5_options = {
    "url": "jdbc:mysql://<jdbc-host-name>:3306/db",
    "dbtable": "test",
    "user": "admin",
    "password": "pwd"}

connection_mysql8_options = {
    "url": "jdbc:mysql://<jdbc-host-name>:3306/db",
    "dbtable": "test",
    "user": "admin",
    "password": "pwd",
    "customJdbcDriverS3Path": "s3://DOC-EXAMPLE-BUCKET/mysql-connector-
java-8.0.17.jar",
    "customJdbcDriverClassName": "com.mysql.cj.jdbc.Driver"}

connection_oracle11_options = {
    "url": "jdbc:oracle:thin:@//<jdbc-host-name>:1521/ORCL",
    "dbtable": "test",
    "user": "admin",
    "password": "pwd"}

connection_oracle18_options = {
    "url": "jdbc:oracle:thin:@//<jdbc-host-name>:1521/ORCL",
    "dbtable": "test",
    "user": "admin",
    "password": "pwd",
    "customJdbcDriverS3Path": "s3://DOC-EXAMPLE-BUCKET/ojdbc10.jar",
    "customJdbcDriverClassName": "oracle.jdbc.OracleDriver"}
```

```
# Read from JDBC databases with custom driver
df_mysql8 = glueContext.create_dynamic_frame.from_options(connection_type="mysql",

 connection_options=connection_mysql8_options)

# Read DynamicFrame from MySQL 5 and write to MySQL 8
df_mysql5 = glueContext.create_dynamic_frame.from_options(connection_type="mysql",

 connection_options=connection_mysql5_options)
glueContext.write_from_options(frame_or_dfc=df_mysql5, connection_type="mysql",
                               connection_options=connection_mysql8_options)

# Read DynamicFrame from Oracle 11 and write to Oracle 18
df_oracle11 =
 glueContext.create_dynamic_frame.from_options(connection_type="oracle",

 connection_options=connection_oracle11_options)
glueContext.write_from_options(frame_or_dfc=df_oracle11, connection_type="oracle",
                               connection_options=connection_oracle18_options)
```

Scala

```
import com.amazonaws.services.glue.GlueContext
import com.amazonaws.services.glue.MappingSpec
import com.amazonaws.services.glue.errors.CallSite
import com.amazonaws.services.glue.util.GlueArgParser
import com.amazonaws.services.glue.util.Job
import com.amazonaws.services.glue.util.JsonOptions
import com.amazonaws.services.glue.DynamicFrame
import org.apache.spark.SparkContext
import scala.collection.JavaConverters._


object GlueApp {
  val MYSQL_5_URI: String = "jdbc:mysql://<jdbc-host-name>:3306/db"
  val MYSQL_8_URI: String = "jdbc:mysql://<jdbc-host-name>:3306/db"
  val ORACLE_11_URI: String = "jdbc:oracle:thin:@//<jdbc-host-name>:1521/ORCL"
  val ORACLE_18_URI: String = "jdbc:oracle:thin:@//<jdbc-host-name>:1521/ORCL"

  // Construct JDBC connection options
  lazy val mysql5JsonOption = jsonOptions(MYSQL_5_URI)
```

```
  lazy val mysql8JsonOption = customJDBCDriverJsonOptions(MYSQL_8_URI, "s3://DOC-
EXAMPLE-BUCKET/mysql-connector-java-8.0.17.jar", "com.mysql.cj.jdbc.Driver")
  lazy val oracle11JsonOption = jsonOptions(ORACLE_11_URI)
  lazy val oracle18JsonOption = customJDBCDriverJsonOptions(ORACLE_18_URI, "s3://
DOC-EXAMPLE-BUCKET/ojdbc10.jar", "oracle.jdbc.OracleDriver")

  def main(sysArgs: Array[String]): Unit = {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)
    val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB_NAME").toArray)
    Job.init(args("JOB_NAME"), glueContext, args.asJava)

    // Read from JDBC database with custom driver
    val df_mysql8: DynamicFrame = glueContext.getSource("mysql",
 mysql8JsonOption).getDynamicFrame()

    // Read DynamicFrame from MySQL 5 and write to MySQL 8
    val df_mysql5: DynamicFrame = glueContext.getSource("mysql",
 mysql5JsonOption).getDynamicFrame()
    glueContext.getSink("mysql", mysql8JsonOption).writeDynamicFrame(df_mysql5)

    // Read DynamicFrame from Oracle 11 and write to Oracle 18
    val df_oracle11: DynamicFrame = glueContext.getSource("oracle",
 oracle11JsonOption).getDynamicFrame()
    glueContext.getSink("oracle", oracle18JsonOption).writeDynamicFrame(df_oracle11)

    Job.commit()
  }

  private def jsonOptions(url: String): JsonOptions = {
    new JsonOptions(
      s"""{"url": "${url}",
         |"dbtable":"test",
         |"user": "admin",
         |"password": "pwd"}""".stripMargin)
  }

  private def customJDBCDriverJsonOptions(url: String, customJdbcDriverS3Path:
 String, customJdbcDriverClassName: String): JsonOptions = {
    new JsonOptions(
      s"""{"url": "${url}",
         |"dbtable":"test",
         |"user": "admin",
         |"password": "pwd",
```

```
        |"customJdbcDriverS3Path": "${customJdbcDriverS3Path}",
        |"customJdbcDriverClassName" :
 "${customJdbcDriverClassName}"}""".stripMargin)
  }
}
```

Lectura desde tablas de JDBC en paralelo

Puede establecer las propiedades de su tabla de JDBC para permitir a AWS Glue leer datos en paralelo. Cuando establece determinadas propiedades, le indica a AWS Glue que ejecute consultas SQL en paralelo en particiones lógicas de sus datos. Puede controlar la partición mediante el establecimiento de un campo hash o una expresión hash. También puede controlar el número de lecturas en paralelo que se utilizan para obtener acceso a los datos.

La lectura paralela de tablas JDBC es una técnica de optimización que puede mejorar el rendimiento. Para obtener más información sobre el proceso de identificación de cuándo esta técnica es adecuada, consulte Reducir la cantidad de datos escaneados en la guía de Prácticas recomendadas para el ajuste del rendimiento de AWS Glue para trabajos de Apache Spark en las Recomendaciones de AWS.

Para habilitar las lecturas en paralelo, puede establecer pares clave-valor en el campo de parámetros de la estructura de tabla. Utilice la notación JSON para establecer un valor del campo de parámetros de la tabla. Para obtener más información acerca de la edición de las propiedades de una tabla, consulte Visualización y edición de los detalles de la tabla. También puede habilitar lecturas en paralelo cuando llama a los métodos de ETL (extraer, transformar y cargar) `create_dynamic_frame_from_options` y `create_dynamic_frame_from_catalog`. Para obtener más información sobre cómo especificar opciones en estos métodos, consulte from_options y from_catalog.

Puede utilizar este método para tablas de JDBC, es decir, la mayoría de tablas de datos cuya datos base es un almacén de datos de JDBC. Estas propiedades se ignoran cuando se leen las tablas de Amazon Redshift y Amazon S3.

hashfield

> Establezca `hashfield` como el nombre de una columna de la tabla de JDBC que se va a utilizar para dividir los datos en particiones. Para obtener los mejores resultados, esta columna debe tener una distribución uniforme de valores para distribuir los datos entre particiones. Esta columna pueden ser de cualquier tipo de datos. AWS Glue genera consultas no solapadas que se ejecutan

en paralelo para leer los datos particionados por esta columna. Por ejemplo, si los datos están distribuidos de manera uniforme por mes, puede utilizar la columna `month` para leer cada mes de datos en paralelo.

```
'hashfield': 'month'
```

AWS Glue crea una consulta para aplicar la función hash en el valor del campo y convertirlo en un número de partición, y ejecuta la consulta para todas las particiones en paralelo. Para utilizar su propia consulta para dividir la lectura de una tabla, proporcione `hashexpression` en lugar de `hashfield`.

hashexpression

Establezca `hashexpression` como una expresión SQL (que cumpla la gramática del motor de base de datos JDBC) que devuelve un número entero. Una expresión simple es el nombre de cualquier columna numérica en la tabla. AWS Glue genera consultas SQL para leer los datos de JDBC en paralelo mediante `hashexpression` en la cláusula `WHERE` para dividir los datos.

Por ejemplo, utilice la columna numérica `customerID` para leer los datos particionados por un número de cliente.

```
'hashexpression': 'customerID'
```

Para que AWS Glue controle la partición, proporcione `hashfield` en lugar de `hashexpression`.

hashpartitions

Establezca `hashpartitions` como el número de lecturas en paralelo de la tabla de JDBC. Si no se define esta propiedad, el valor predeterminado es 7.

Por ejemplo, establezca el número de lecturas en paralelo en `5`, de modo que AWS Glue lea los datos con cinco consultas (o menos).

```
'hashpartitions': '5'
```

Configuración de una VPC de Amazon para conexiones JDBC a los almacenes de datos de Amazon RDS de AWS Glue

Al utilizar JDBC para conectar las bases de datos en Amazon RDS, se necesitará realizar una configuración adicional. Para habilitar la comunicación entre los componentes de AWS Glue y Amazon RDS, debe configurar el acceso a los almacenes de datos de Amazon RDS a la VPC de Amazon. Para que AWS Glue pueda comunicarse entre sus componentes, especifique un grupo de seguridad con una regla de entrada con autorreferencia para todos los puertos TCP. Al crear una regla con autorreferencia, se puede restringir el origen al mismo grupo de seguridad en la VPC. Una regla con autorreferencia no abrirá la VPC en todas las redes. Puede que el grupo de seguridad predeterminado de la VPC ya tenga una regla de entrada con autorreferencia para ALL Traffic.

Para configurar el acceso entre los almacenes de datos de AWS Glue y Amazon RDS

1. Inicie sesión en la consola de Amazon RDS AWS Management Console y ábrala en https://console.aws.amazon.com/rds/.
2. En la consola de Amazon RDS, identifique los grupos de seguridad que se utilizan para controlar el acceso a la base de datos de Amazon RDS.

   En el panel de navegación a la izquierda, seleccione Bases de datos y, a continuación, seleccione la instancia a la que desea conectarse de la lista del panel principal.

   En la página de detalles de la base de datos, busque Grupos de seguridad de VPC dentro de la pestaña Conectividad y seguridad.
3. Según la arquitectura de red, identifique qué grupo de seguridad asociado es mejor modificar para permitir el acceso al servicio AWS Glue. Guarde su nombre *database-security-group*para consultarlo en el futuro. Si no hay un grupo de seguridad adecuado, siga las instrucciones para brindar acceso a la instancia de base de datos en la VPC con la creación de un grupo de seguridad en la documentación de Amazon RDS.
4. Inicie sesión en la consola de Amazon VPC AWS Management Console y ábrala en https://console.aws.amazon.com/vpc/.
5. En la consola de Amazon VPC, identifique cómo realizar la actualización. *database-security-group*

   En el panel de navegación izquierdo, elija Grupos de seguridad y, a continuación, seleccione una opción *database-security-group*de la lista del panel principal.
6. Identifique el ID del grupo de seguridad para *database-security-group*, *database-sg-id*. Guárdelo para consultarlo en el futuro.

En la página de detalles del grupo de seguridad, busque ID del grupo de seguridad.

7. Modifique las reglas de entrada y añada una regla de autorreferencia para *database-security-group*permitir que AWS Glue los componentes se comuniquen. En concreto, añada o confirme que hay una regla en la que el tipo es**All TCP**, el protocolo es**TCP**, el rango de puertos incluye todos los puertos y el origen es. ***database-sg-id*** Compruebe que el grupo de seguridad que introduzca en Origen sea el mismo que el grupo de seguridad que está editando.

   En la página de detalles del grupo de seguridad, seleccione Editar reglas de entrada.

   La regla de entrada tiene un aspecto similar al siguiente:

Tipo	Protocolo	Intervalo de puertos	Origen
Todos los TCP	TCP	0–65535	*database-sg-id*

8. Agregue reglas para el tráfico saliente.

   En la página de detalles del grupo de seguridad, seleccione Editar reglas de salida.

   En caso de que el grupo de seguridad permita todo el tráfico de salida, no se necesitan reglas independientes. Por ejemplo:

Tipo	Protocolo	Rango de puerto	Destino
All Traffic	ALL	ALL	0.0.0.0/0

   Si la arquitectura de red está diseñada para restringir el tráfico de salida, cree las siguientes reglas salientes:

   Cree una regla de autorreferencia en la que el tipo seaAll TCP, el protocolo seaTCP, el rango de puertos incluya todos los puertos y el destino sea. *database-sg-id* Compruebe que el grupo de seguridad que introduzca en Destino sea el mismo que el grupo de seguridad que está editando.

   En caso de que se utilice un punto de conexión de VPC de Amazon S3, agregue una regla HTTPS para permitir el tráfico desde la VPC a Amazon S3. Cree una regla en la que el tipo seaHTTPS, el protocolo seaTCP, el rango de puertos sea 443 y el destino sea el ID de la lista

de prefijos administrada para el punto de enlace de la puerta de enlace Amazon *S3, s3-prefix-list-id*. Para obtener más información sobre las listas de prefijos y los puntos de conexión de la puerta de enlace de Amazon S3, consulte Puntos de conexión de la puerta de enlace de Amazon S3 en la documentación de la VPC de Amazon.

Por ejemplo:

Tipo	Protocolo	Rango de puerto	Destino
Todos los TCP	TCP	0–65535	*database-sg-id*
HTTPS	TCP	443	*s3- prefix-list-id*

Conexiones a MongoDB

Puede usar AWS Glue for Spark para leer y escribir en tablas de MongoDB y MongoDB Atlas en AWS Glue 4.0 y versiones posteriores. Puede conectarse a MongoDB con las credenciales de nombre de usuario y contraseña almacenadas en AWS Secrets Manager a través de una conexión de AWS Glue.

Para obtener más información sobre MongoDB, consulte la documentación de MongoDB.

Configuración de las conexiones MongoDB

Para conectarte a MongoDB desde AWS Glue, necesitará sus credenciales de MongoDB, *mongodbUser* y *mongodbPass*.

Para conectarse a MongoDB desde AWS Glue, es posible que necesite algunos requisitos previos:

- Si su instancia de MongoDB está en una VPC de Amazon, configure Amazon VPC para permitir que su trabajo de AWS Glue se comunique con la instancia de MongoDB sin que el tráfico atraviese la Internet pública.

  En Amazon VPC, identifique o cree una VPC, una subred y un grupo de seguridad que AWS Glue utilizará al ejecutar el trabajo. Además, debe asegurarse de que Amazon VPC esté configurada para permitir el tráfico de red entre su instancia de MongoDB y esta ubicación. Según el diseño de la red, esto puede requerir cambios en las reglas de los grupos de seguridad, las ACL de red, las puertas de enlace de NAT y las conexiones entre pares.

A continuación, puede proceder a configurar AWS Glue para su uso con MongoDB.

Para configurar una conexión a MongoDB:

1. Si lo desea, en AWS Secrets Manager, cree un secreto con sus credenciales de MongoDB. Para crear un secreto en Secrets Manager, siga el tutorial disponible en Crear un secreto AWS Secrets Manager en la documentación AWS Secrets Manager. Después de crear el secreto, conserve el nombre secreto, *secretName*, para el siguiente paso.

   - Al seleccionar pares clave/valor, genere un par para la clave `username` con el valor *mongodbUser*.

     Al seleccionar pares clave/valor, genere un par para la clave `password` con el valor *mongodbPass*.

2. En la consola de AWS Glue, genere una conexión mediante los pasos que se indican en the section called “Adición de una conexión de AWS Glue”. Tras crear la conexión, conserve el nombre de la conexión, *connectionName*, para el uso futuro en AWS Glue.

   - Al seleccionar un tipo de conexión, seleccione MongoDB o MongoDB Atlas.
   - Al seleccionar la URL de MongoDB o la URL de MongoDB Atlas, proporciona el nombre de host de la instancia de MongoDB.

     Se proporciona una URL de MongoDB en este formato `mongodb://`*mongoHost*`:`*mongoPort*`/`*mongoDBname*.

     Se proporciona una URL de MongoDB Atlas en este formato `mongodb +srv://`*mongoHost*`:`*mongoPort*`/`*mongoDBname*.

     Proporcionar la base de datos predeterminada para la conexión, *mongoDBname* es opcional.
   - Si eligió crear un secreto de Secrets Manager, elija el tipo de credencial AWS Secrets Manager.

     Luego, en AWS Secret, ingrese *secretName*.
   - Si decide proporcionar el nombre de usuario y la contraseña, proporcione *mongoDBuser* y *mongodbPass*.

3. En las siguientes situaciones, es posible que necesite una configuración adicional:

   - Para instancias de MongoDB alojadas AWS en una VPC de Amazon

- Deberá proporcionar la información de conexión de Amazon VPC a la conexión AWS Glue que define sus credenciales de seguridad de MongoDB. Al crear o actualizar la conexión, configure los VPC, Subred y los grupos de seguridad en Opciones de red.

Tras crear una conexión AWS Glue MongoDB, tendrá que realizar las siguientes acciones antes de llamar a su método de conexión:

- Si ha decidido crear un secreto de Secrets Manager, conceda permiso al rol de IAM asociado a su trabajo de AWS Glue para leer *`secretName`*.
- En la configuración del trabajo de Glue AWS, proporcione *`connectionName`* como una conexión de red adicional.

Para usar su conexión AWS Glue MongoDB en AWS Glue for Spark, proporcione la opción `connectionName` en su llamada al método de conexión. Como alternativa, puede seguir los pasos que se describen en the section called “Integración con MongoDB” para utilizar la conexión junto con el catálogo de datos de AWS Glue.

Lectura de MongoDB mediante una conexión a AWS Glue

Requisitos previos:

- Una colección de MongoDB de la que quiera leer. Necesitará información de identificación para la colección.

  Una colección MongoDB se identifica mediante un nombre de base de datos y un nombre de colección, *`mongodbName`*, *`mongodbCollection`*.
- Una conexión AWS Glue MongoDB configurada para proporcionar información de autenticación. Complete los pasos del procedimiento anterior, Para configurar una conexión a MongoDB para configurar su información de autenticación. Necesitará el nombre de la conexión de AWS Glue, *`connectionName`*.

Por ejemplo:

```
mongodb_read = glueContext.create_dynamic_frame.from_options(
    connection_type="mongodb",
    connection_options={
        "connectionName": "connectionName",
        "database": "mongodbName",
```

```
        "collection": "mongodbCollection",
        "partitioner":
 "com.mongodb.spark.sql.connector.read.partitioner.SinglePartitionPartitioner",
        "partitionerOptions.partitionSizeMB": "10",
        "partitionerOptions.partitionKey": "_id",
        "disableUpdateUri": "false",
    }
)
```

Escribir en tablas de MongoDB

En este ejemplo se escribe información de un DynamicFrame existente, *dynamicFrame* en MongoDB.

Requisitos previos:

- Una colección de MongoDB a la que desearía escribir. Necesitará información de identificación para la colección.

  Una colección MongoDB se identifica mediante un nombre de base de datos y un nombre de colección, *mongodbName*, *mongodbCollection*.
- Una conexión AWS Glue MongoDB configurada para proporcionar información de autenticación. Complete los pasos del procedimiento anterior, Para configurar una conexión a MongoDB para configurar su información de autenticación. Necesitará el nombre de la conexión de AWS Glue, *connectionName*.

Por ejemplo:

```
glueContext.write_dynamic_frame.from_options(
    frame=dynamicFrame,
    connection_type="mongodb",
    connection_options={
        "connectionName": "connectionName",
        "database": "mongodbName",
        "collection": "mongodbCollection",
        "disableUpdateUri": "false",
        "retryWrites": "false",
    },
)
```

Leer y escribir en tablas de MongoDB

En este ejemplo se escribe información de un DynamicFrame existente, *dynamicFrame* en MongoDB.

Requisitos previos:

- Una colección de MongoDB de la que quiera leer. Necesitará información de identificación para la colección.

  Una colección de MongoDB a la que desearía escribir. Necesitará información de identificación para la colección.

  Una colección MongoDB se identifica mediante un nombre de base de datos y un nombre de colección, *mongodbName*, *mongodbCollection*.
- Información de autenticación de MongoDB, *mongodbUser* y *mongodbPassword*.

Por ejemplo:

Python

```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext, SparkConf
from awsglue.context import GlueContext
from awsglue.job import Job
import time

## @params: [JOB_NAME]
args = getResolvedOptions(sys.argv, ['JOB_NAME'])

sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session

job = Job(glueContext)
job.init(args['JOB_NAME'], args)

output_path = "s3://some_bucket/output/" + str(time.time()) + "/"
mongo_uri = "mongodb://<mongo-instanced-ip-address>:27017"
mongo_ssl_uri = "mongodb://<mongo-instanced-ip-address>:27017"
```

```
write_uri = "mongodb://<mongo-instanced-ip-address>:27017"

read_mongo_options = {
    "uri": mongo_uri,
    "database": "mongodbName",
    "collection": "mongodbCollection",
    "username": "mongodbUsername",
    "password": "mongodbPassword",
    "partitioner": "MongoSamplePartitioner",
    "partitionerOptions.partitionSizeMB": "10",
    "partitionerOptions.partitionKey": "_id"}

ssl_mongo_options = {
    "uri": mongo_ssl_uri,
    "database": "mongodbName",
    "collection": "mongodbCollection",
    "ssl": "true",
    "ssl.domain_match": "false"
}

write_mongo_options = {
    "uri": write_uri,
    "database": "mongodbName",
    "collection": "mongodbCollection",
    "username": "mongodbUsername",
    "password": "mongodbPassword",
}

# Get DynamicFrame from MongoDB
dynamic_frame =
 glueContext.create_dynamic_frame.from_options(connection_type="mongodb",

 connection_options=read_mongo_options)

# Write DynamicFrame to MongoDB
glueContext.write_dynamic_frame.from_options(dynamicFrame,
 connection_type="mongodb", connection_options=write_mongo_options)

job.commit()
```

Scala

```
import com.amazonaws.services.glue.GlueContext
```

```
import com.amazonaws.services.glue.MappingSpec
import com.amazonaws.services.glue.errors.CallSite
import com.amazonaws.services.glue.util.GlueArgParser
import com.amazonaws.services.glue.util.Job
import com.amazonaws.services.glue.util.JsonOptions
import com.amazonaws.services.glue.DynamicFrame
import org.apache.spark.SparkContext
import scala.collection.JavaConverters._

object GlueApp {
  val DEFAULT_URI: String = "mongodb://<mongo-instanced-ip-address>:27017"
  val WRITE_URI: String = "mongodb://<mongo-instanced-ip-address>:27017"
  lazy val defaultJsonOption = jsonOptions(DEFAULT_URI)
  lazy val writeJsonOption = jsonOptions(WRITE_URI)
  def main(sysArgs: Array[String]): Unit = {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)
    val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB_NAME").toArray)
    Job.init(args("JOB_NAME"), glueContext, args.asJava)

    // Get DynamicFrame from MongoDB
    val dynamicFrame: DynamicFrame = glueContext.getSource("mongodb",
 defaultJsonOption).getDynamicFrame()

    // Write DynamicFrame to MongoDB
    glueContext.getSink("mongodb", writeJsonOption).writeDynamicFrame(dynamicFrame)

    Job.commit()
  }

  private def jsonOptions(uri: String): JsonOptions = {
    new JsonOptions(
      s"""{"uri": "${uri}",
         |"database":"mongodbName",
         |"collection":"mongodbCollection",
         |"username": "mongodbUsername",
         |"password": "mongodbPassword",
         |"ssl":"true",
         |"ssl.domain_match":"false",
         |"partitioner": "MongoSamplePartitioner",
         |"partitionerOptions.partitionSizeMB": "10",
         |"partitionerOptions.partitionKey": "_id"}""".stripMargin)
  }
```

```
}
```

Referencia de opciones de conexión de MongoDB

Designa una conexión a MongoDB. Las opciones de conexión son distintas entre una conexión de origen y una conexión de receptor.

Estas propiedades de conexión se comparten entre las conexiones de origen y de receptor:

- `connectionName`: se utiliza para lectura/escritura. El nombre de una conexión AWS Glue MongoDB configurada para proporcionar información de autenticación y red a su método de conexión. Cuando se configura una conexión AWS Glue como se describe en la sección anterior, the section called “Configuración de MongoDB”, el suministro `connectionName` sustituirá la necesidad de proporcionar las opciones de conexión `"uri"`, `"username"` y `"password"`.
- `"uri"`: (obligatorio) el host de MongoDB del que se va a leer, con formato `mongodb://<host>:<port>`. Se utiliza en las versiones de AWS Glue anteriores a AWS Glue 4.0.
- `"connection.uri"`: (obligatorio) el host de MongoDB del que se va a leer, con formato `mongodb://<host>:<port>`. Es compatible con la versión AWS Glue 4.0 y posteriores.
- `"username"`: (obligatorio) el nombre de usuario de MongoDB.
- `"password"`: (obligatorio) la contraseña de MongoDB.
- `"database"`: (obligatorio) la base de datos de MongoDB de la que se va a leer. Esta opción también se puede transferir a `additional_options` al llamar a `glue_context.create_dynamic_frame_from_catalog` en su script de trabajo.
- `"collection"`: (obligatorio) la colección de MongoDB de la que se va a leer. Esta opción también se puede transferir a `additional_options` al llamar a `glue_context.create_dynamic_frame_from_catalog` en su script de trabajo.

"connectionType": "mongodb" como origen

Utilice las siguientes opciones de conexión con `"connectionType": "mongodb"` como origen:

- `"ssl"`: (opcional) si es `true`, inicia una conexión SSL. El valor predeterminado es `false`.
- `"ssl.domain_match"`: (opcional) si es `true` y `ssl` es `true`, se realiza la comprobación de coincidencia de dominio. El valor predeterminado es `true`.
- `"batchSize"`: (opcional): el número de documentos que se deben devolver por lote, que se utilizan dentro del cursor de lotes internos.

- `"partitioner"`: (opcional): el nombre de la clase del particionador para leer los datos de entrada de MongoDB. El conector proporciona los siguientes particionadores:
  - `MongoDefaultPartitioner` (predeterminado) (No compatible con AWS Glue 4.0)
  - `MongoSamplePartitioner` (Requiere MongoDB 3.2 o posterior) (No es compatible con AWS Glue 4.0)
  - `MongoShardedPartitioner` (No es compatible con AWS Glue 4.0)
  - `MongoSplitVectorPartitioner` (No es compatible con AWS Glue 4.0)
  - `MongoPaginateByCountPartitioner` (No es compatible con AWS Glue 4.0)
  - `MongoPaginateBySizePartitioner` (No es compatible con AWS Glue 4.0)
  - `com.mongodb.spark.sql.connector.read.partitioner.SinglePartitionPartitioner`
  - `com.mongodb.spark.sql.connector.read.partitioner.ShardedPartitioner`
  - `com.mongodb.spark.sql.connector.read.partitioner.PaginateIntoPartitionsPartit`
- `"partitionerOptions"` (opcional): opciones para el particionador designado. Se admiten las siguientes opciones para cada particionador:
  - `MongoSamplePartitioner`: `partitionKey`, `partitionSizeMB`, `samplesPerPartition`
  - `MongoShardedPartitioner`: `shardkey`
  - `MongoSplitVectorPartitioner`: `partitionKey`, `partitionSizeMB`
  - `MongoPaginateByCountPartitioner`: `partitionKey`, `numberOfPartitions`
  - `MongoPaginateBySizePartitioner`: `partitionKey`, `partitionSizeMB`

  Para obtener más información acerca de estas opciones, consulte Partitioner Configuration en la documentación de MongoDB.

"connectionType": "mongodb" como receptor

Utilice las siguientes opciones de conexión con `"connectionType": "mongodb"` como receptor:

- `"ssl"`: (opcional) si es `true`, inicia una conexión SSL. El valor predeterminado es `false`.
- `"ssl.domain_match"`: (opcional) si es `true` y `ssl` es `true`, se realiza la comprobación de coincidencia de dominio. El valor predeterminado es `true`.
- `"extendedBsonTypes"`: (opcional) si es `true`, habilitan los tipos de BSON ampliados al escribir datos en MongoDB. El valor predeterminado es `true`.

- `"replaceDocument"`: (opcional) si es `true`, reemplaza todo el documento al guardar conjuntos de datos que contienen un campo `_id`. Si es `false`, solo se actualizan los campos del documento que coinciden con los campos del conjunto de datos. El valor predeterminado es `true`.
- `"maxBatchSize"`: (opcional): el tamaño máximo del lote para operaciones en bloque al guardar datos. El valor predeterminado es 512.
- `"retryWrites"`: (Opcional): reintenta automáticamente determinadas operaciones de escritura una sola vez si AWS Glue detecta un error de red.

Conexiones SAP HANA

Puede usar AWS Glue for Spark para leer y escribir en tablas de SAP HANA en AWS Glue 4.0 y versiones posteriores. Puede definir qué leer desde SAP HANA con una consulta SQL. Se conecta a SAP HANA con las credenciales JDBC almacenadas en AWS Secrets Manager a través de una conexión de AWS Glue SAP HANA.

Para obtener más información sobre SAP HANA JDBC, consulte la documentación de SAP HANA.

Configuración de conexiones SAP HANA

Para conectarse a SAP HANA desde AWS Glue, tendrá que crear y almacenar las credenciales de SAP HANA en un AWS Secrets Manager secreto y, a continuación, asociar ese secreto a una conexión de SAP HANA AWS Glue. Deberá configurar la conectividad de red entre su servicio SAP HANA y AWS Glue.

Para conectarse a SAP HANA, es posible que necesite algunos requisitos previos:

- Si su servicio SAP HANA está en una VPC de Amazon, configure Amazon VPC para permitir que su trabajo de AWS Glue comunique con el servicio SAP HANA sin que el tráfico atraviese la Internet pública.

  En Amazon VPC, identifique o cree una VPC, una subred y un grupo de seguridad que AWS Glue utilizará al ejecutar el trabajo. Además, debe asegurarse de que Amazon VPC esté configurada para permitir el tráfico de red entre su punto de conexión SAP HANA y esta ubicación. Su trabajo deberá establecer una conexión TCP con su puerto JDBC de SAP HANA. Para obtener más información sobre los puertos de SAP HANA, consulte la documentación de SAP HANA. Según el diseño de la red, esto puede requerir cambios en las reglas de los grupos de seguridad, las ACL de red, las puertas de enlace de NAT y las conexiones entre pares.

- No hay requisitos previos adicionales si se puede acceder a su punto de conexión de SAP HANA a través de Internet.

Para configurar una conexión a SAP HANA:

1. En AWS Secrets Manager, cree un secreto con sus credenciales de SAP HANA. Para crear un secreto en Secrets Manager, siga el tutorial disponible en Crear un secreto AWS Secrets Manager en la documentación AWS Secrets Manager. Después de crear el secreto, conserve el nombre secreto, *secretName*, para el siguiente paso.

   - Al seleccionar pares clave/valor, genere un par para la clave `user` con el valor *saphanaUsername*.
   - Al seleccionar pares clave/valor, genere un par para la clave `password` con el valor *saphanaPassword*.

2. En la consola de AWS Glue, genere una conexión mediante los pasos que se indican en the section called “Adición de una conexión de AWS Glue”. Tras crear la conexión, conserve el nombre de la conexión, *connectionName*, para el uso futuro en AWS Glue.

   - Al seleccionar un tipo de conexión, seleccione SAP HANA.
   - Al proporcionar la URL de SAP HANA, proporcione la URL de su instancia.

     Las URL de JDBC de SAP HANA tienen el formato `jdbc:sap://`*saphanaHostname*`:`*saphanaPort*`/?`*databaseName*`=`*saphanaDBname*`,`*Paramete*

     AWS Glue requiere los siguientes parámetros de URL de JDBC:

     - `databaseName` — Una base de datos predeterminada en SAP HANA a la que conectarse.
   - Al seleccionar un secreto AWS, proporcione un *secretName*.

Tras crear una conexión con AWS Glue SAP HANA, deberá realizar los siguientes pasos antes de ejecutar su trabajo de AWS Glue:

- Otorgue al rol de IAM asociado al permiso de su trabajo de AWS Glue para leer *secretName*.
- En la configuración del trabajo de Glue AWS, proporcione *connectionName* como una conexión de red adicional.

Lectura de tablas de SAP HANA

Requisitos previos:

- Una tabla de SAP HANA desde la cual le gustaría leer. Necesitará información de la identificación para la tabla.

  Se puede especificar una tabla con un nombre de tabla y un nombre de esquema de SAP HANA, en la forma *schemaName*.*tableName*. El nombre del esquema y el separador "." no son necesarios si la tabla está en el esquema predeterminado, "public". Esto se llama *tableIdentifier*. Tenga en cuenta que la base de datos se proporciona como un parámetro de URL de JDBC en `connectionName`.
- Una conexión AWS Glue SAP HANA configurada para proporcionar información de autenticación. Complete los pasos del procedimiento anterior, Para configurar una conexión a SAP HANA y configurar su información de autenticación. Necesitará el nombre de la conexión de AWS Glue, *connectionName*.

Por ejemplo:

```
saphana_read_table = glueContext.create_dynamic_frame.from_options(
    connection_type="saphana",
    connection_options={
        "connectionName": "connectionName",
        "dbtable": "tableIdentifier",
    }
)
```

También puede proporcionar una consulta SELECT SQL para filtrar los resultados devueltos a su DynamicFrame. Deberá configurar `query`.

Por ejemplo:

```
saphana_read_query = glueContext.create_dynamic_frame.from_options(
    connection_type="saphana",
    connection_options={
        "connectionName": "connectionName",
        "query": "query"
    }
)
```

Escribir en tablas de SAP HANA

Este ejemplo escribe información de un DynamicFrame existente, *dynamicFrame* en SAP HANA. Si la tabla ya contiene información, AWS Glue generará un error.

Requisitos previos:

- Una tabla de SAP HANA en la cual le gustaría escribir.

  Se puede especificar una tabla con un nombre de tabla y un nombre de esquema de SAP HANA, en la forma *schemaName*.*tableName*. El nombre del esquema y el separador "." no son necesarios si la tabla está en el esquema predeterminado, "public". Esto se llama *tableIdentifier*. Tenga en cuenta que la base de datos se proporciona como un parámetro de URL de JDBC en `connectionName`.
- Información de autenticación de SAP HANA. Complete los pasos del procedimiento anterior, Para configurar una conexión a SAP HANA y configurar su información de autenticación. Necesitará el nombre de la conexión de AWS Glue, *connectionName*.

Por ejemplo:

```
options = {
    "connectionName": "connectionName",
    "dbtable": 'tableIdentifier'
}

    saphana_write = glueContext.write_dynamic_frame.from_options(
        frame=dynamicFrame,
        connection_type="saphana",
        connection_options=options
)
```

Referencia de opciones de conexión de SAP HANA

- `connectionName`: obligatorio. Se utiliza para lectura/escritura. El nombre de una conexión de AWS Glue SAP HANA configurada para proporcionar información de autenticación y red a su método de conexión.
- `databaseName`: se utiliza para lectura/escritura. Valores válidos: nombres de las bases de datos en SAP HANA. Nombre de la base de datos a la que se va a conectar.

- `dbtable` — Necesario para escribir, obligatorio para leer a menos que se proporcione `query`. Se utiliza para lectura/escritura. Valores válidos: contenido de una cláusula SQL FROM de SAP HANA. Identifica una tabla en SAP HANA a la que conectarse. También puede proporcionar un SQL distinto del nombre de una tabla, como una subconsulta. Para obtener más información, consulte la cláusula From en la documentación de SAP HANA.
- `query`: se utiliza para leer. Una consulta de SAP HANA SQL SELECT que define lo que se debe recuperar al leer información de SAP HANA.

Conexiones Snowflake

Puede usar Glue for Spark AWS para leer y escribir en tablas de Snowflake en Glue 4.0 AWS y versiones posteriores. Puede leer desde Snowflake con una consulta SQL. Puede conectarse a Snowflake mediante un usuario y una contraseña. Puede consultar las credenciales de Snowflake almacenadas en AWS Secrets Manager a través del catálogo de datos de Glue AWS. Las credenciales de Snowflake del catálogo de datos para Glue for Spark AWS se almacenan por separado de las credenciales de Snowflake del catálogo de datos para los rastreadores. Debe elegir un tipo de conexión `SNOWFLAKE` y no un tipo de conexión `JDBC` configurado para conectarse a Snowflake.

Para obtener más información acerca de Snowflake, consulte el sitio web de Snowflake. Para obtener más información sobre Snowflake en AWS, consulte Snowflake Data Warehouse en Amazon Web Services.

Configuración de las conexiones de Snowflake

No hay requisitos previos AWS para conectarse a las bases de datos de Snowflake disponibles a través de Internet.

Si lo desea, puede realizar la siguiente configuración para gestionar las credenciales de conexión con Glue AWS.

Para gestionar sus credenciales de conexión con Glue AWS

1. En Snowflake, genere un usuario *`snowflakeUser`* y una contraseña *`snowflakePassword`*.
2. En AWS Secrets Manager, cree un secreto con sus credenciales de Snowflake. Para crear un secreto en Secrets Manager, siga el tutorial disponible en Crear un secreto AWS Secrets Manager en la documentación AWS Secrets Manager. Después de crear el secreto, conserve el nombre secreto, *`secretName`*, para el siguiente paso.

- Al seleccionar pares clave/valor, cree un par para *snowflakeUser* con la clave `sfUser`.
- Al seleccionar pares clave/valor, cree un par para *snowflakePassword* con la clave `sfPassword`.
- Al seleccionar los pares clave/valor, puede proporcionar su almacén de Snowflake con la clave `sfWarehouse`.

3. En el catálogo de datos de Glue AWS, cree una conexión mediante los pasos que se indican en the section called “Adición de una conexión de AWS Glue”. Tras crear la conexión, conserve el nombre de la conexión, *connectionName*, para el siguiente paso.

   - Al seleccionar un tipo de conexión, seleccione Snowflake.
   - Al seleccionar la URL de Snowflake, proporcione la URL de su instancia de Snowflake. La URL utilizará un nombre de host en el formulario *account_identifier*`.snowflakecomputing.com`.
   - Al seleccionar un secreto AWS, proporcione un *secretName*.

4. En la configuración del trabajo de Glue AWS, proporcione *connectionName* como una conexión de red adicional.

En las siguientes situaciones, es posible que necesite lo siguiente:

- Para Snowflake alojado en AWS en una VPC de Amazon
  - Necesitará la configuración de VPC de Amazon adecuada para Snowflake. Para obtener más información sobre cómo configurar su VPC de Amazon, consulte AWS PrivateLink & Snowflake en la documentación de Snowflake.
  - Necesitará la configuración de VPC de Amazon adecuada para Glue AWS the section called “Puntos de conexión de VPC (AWS PrivateLink)”.
  - Deberá crear una conexión al catálogo de datos de Glue AWS que proporcione la información de conexión de VPC de Amazon (además del identificador de un secreto AWS Secrets Manager que defina sus credenciales de seguridad de Snowflake). Su URL cambiará cuando utilice AWS PrivateLink, tal y como se describe en la documentación de Snowflake vinculada en un artículo anterior.
  - Necesitará que la configuración de su trabajo incluya la conexión de catálogo de datos como una conexión de red adicional.

Lectura de tablas de Snowflake

Requisitos previos: una tabla de Snowflake de la que quiera leer. Necesitará el nombre de la tabla Snowflake, *tableName*. Necesitará la URL de Snowflake*snowflakeUrl*, el nombre de usuario *snowflakeUser* y la contraseña *snowflakePassword*. Si su usuario de Snowflake no tiene establecido un espacio de nombres predeterminado, necesitará el nombre de la base de datos de Snowflake *databaseName* y el nombre del esquema *schemaName*. Además, si su usuario de Snowflake no tiene un conjunto de almacenes predeterminado, necesitará un nombre de almacén *warehouseName*.

Por ejemplo:

Requisitos previos adicionales: complete los pasos para gestionar credenciales de conexión con Glue AWS para configurar *snowflakeUrl*, *snowflakeUsername* y *snowflakePassword*. Para revisar estos pasos, consulte la sección anterior the section called “Configuración de Snowflake”. Para seleccionar con qué conexión de red adicional nos vamos a conectar, utilizaremos el parámetro `connectionName`.

```
snowflake_read = glueContext.create_dynamic_frame.from_options(
  connection_type="snowflake",
  connection_options={
        "connectionName": "connectionName",
        "dbtable": "tableName",
        "sfDatabase": "databaseName",
        "sfSchema": "schemaName",
        "sfWarehouse": "warehouseName",
    }
)
```

Además, puede usar los parámetros `autopushdown` y `query` para leer una parte de una tabla de Snowflake. Esto puede ser considerablemente más eficiente que filtrar los resultados después de haberlos cargado en Spark. Considere un ejemplo en el que todas las ventas se almacenan en la misma tabla, pero solo necesita analizar las ventas de una tienda determinada los días festivos. Si esa información está almacenada en la tabla, podría utilizar la función de inserción de predicados para recuperar los resultados de la siguiente manera:

```
snowflake_node = glueContext.create_dynamic_frame.from_options(
    connection_type="snowflake",
    connection_options={
        "autopushdown": "on",
```

```
        "query": "select * from sales where store='1' and IsHoliday='TRUE'",
        "connectionName": "snowflake-glue-conn",
        "sfDatabase": "databaseName",
        "sfSchema": "schemaName",
        "sfWarehouse": "warehouseName",
    }
)
```

Escribir en tablas de Snowflake

Requisitos previos: una base de datos de Snowflake a la que desearía escribir. Necesitará un nombre de tabla actual o deseado, *tableName*. Necesitará la URL de Snowflake*snowflakeUrl*, el nombre de usuario *snowflakeUser* y la contraseña *snowflakePassword*. Si su usuario de Snowflake no tiene establecido un espacio de nombres predeterminado, necesitará el nombre de la base de datos de Snowflake *databaseName* y el nombre del esquema *schemaName.* Además, si su usuario de Snowflake no tiene un conjunto de almacenes predeterminado, necesitará un nombre de almacén *warehouseName.*

Por ejemplo:

Requisitos previos adicionales: complete los pasos para gestionar credenciales de conexión con Glue AWS para configurar *snowflakeUrl*, *snowflakeUsername* y *snowflakePassword*. Para revisar estos pasos, consulte la sección anterior the section called “Configuración de Snowflake”. Para seleccionar con qué conexión de red adicional nos vamos a conectar, utilizaremos el parámetro `connectionName`.

```
glueContext.write_dynamic_frame.from_options(
    connection_type="snowflake",
    connection_options={
        "connectionName": "connectionName",
        "dbtable": "tableName",
        "sfDatabase": "databaseName",
        "sfSchema": "schemaName",
        "sfWarehouse": "warehouseName",
    },
)
```

Referencia de la opción de conexión Snowflake

El tipo de conexión Snowflake tiene las siguientes opciones de conexión:

Puede recuperar algunos de los parámetros de esta sección desde una conexión de catálogo de datos (`sfUrl`, `sfUser`, `sfPassword`), en cuyo caso no tendrá que proporcionarlos. Puede hacerlo al proporcionar el parámetro `connectionName`.

Puede recuperar algunos de los parámetros de esta sección desde un secreto AWS Secrets Manager (`sfUser`, `sfPassword`), en cuyo caso no tendrá que proporcionarlos. El secreto debe proporcionar el contenido debajo de las claves `sfUser` y `sfPassword`. Puede hacerlo al proporcionar el parámetro `secretId`.

Los siguientes parámetros se utilizan generalmente cuando se conecta a Snowflake.

- `sfDatabase` — Obligatorio si el usuario predeterminado no está configurado en Snowflake. Se utiliza para lectura/escritura. La base de datos que se utilizará en la sesión después de conectarse.
- `sfSchema` — Obligatorio si el usuario predeterminado no está configurado en Snowflake. Se utiliza para lectura/escritura. El esquema que se utilizará en la sesión después de conectarse.
- `sfWarehouse` — Obligatorio si el usuario predeterminado no está configurado en Snowflake. Se utiliza para lectura/escritura. El almacén virtual predeterminado que se utilizará en la sesión después de la conexión.
- `sfRole` — Obligatorio si el usuario predeterminado no está configurado en Snowflake. Se utiliza para lectura/escritura. Rol de seguridad predeterminado que se utilizará en la sesión después de conectarse.
- `sfUrl` — (Obligatorio) Se utiliza para lectura/escritura. Especifica el nombre de host para su cuenta en el siguiente formato: *`account_identifier`*`.snowflakecomputing.com`. Para obtener más información sobre los identificadores de cuenta, consulte los identificadores de cuenta en la documentación de Snowflake.
- `sfUser` — (Obligatorio) Se utiliza para lectura/escritura. Nombre de inicio de sesión del usuario de Snowflake.
- `sfPassword`: (Obligatorio a menos que `pem_private_key` se proporcione) Se utiliza para lectura/escritura. Contraseña del usuario de Snowflake.
- `dbtable` — Obligatorio cuando se trabaja con tablas completas. Se utiliza para lectura/escritura. Nombre de la tabla que se va a leer o de la tabla en la que se escriben los datos. Al leer, se recuperan todas las columnas y registros.
- `pem_private_key`: se utiliza para lectura/escritura. Cadena de clave privada codificada en b64 sin cifrar. La clave privada para el usuario de Snowflake. Es habitual copiarla de un archivo PEM. Para obtener más información, consulte Key-pair authentication and key-pair rotation en la documentación de Snowflake.

- `query` — Obligatorio cuando se lee con una consulta. Se usa para leer. La consulta exacta (instrucción `SELECT`) que se va a ejecutar

Las siguientes opciones se utilizan para configurar comportamientos específicos durante el proceso de conexión a Snowflake.

- `preactions` — Se utiliza para lectura/escritura. Valores válidos: lista de declaraciones SQL separadas por punto y coma como cadena. Las declaraciones SQL se ejecutan antes de que los datos se transfieran entre Glue AWS y Snowflake. Si una instrucción contiene `%s`, el `%s` se sustituye por el nombre de la tabla a la que se hace referencia para la operación.
- `postactions` — Se utiliza para lectura/escritura. Las declaraciones SQL se ejecutan después de transferir los datos entre Glue AWS y Snowflake. Si una instrucción contiene `%s`, el `%s` se sustituye por el nombre de la tabla a la que se hace referencia para la operación.
- `autopushdown` — Predeterminado: `"on"`. Valores válidos: `"on"`, `"off"`. Este parámetro controla si la inserción de visualización automática de consultas está habilitada. Si la función pushdown está habilitada, cuando se ejecuta una consulta en Spark, si parte de la consulta se puede "enviar" al servidor de Snowflake, se empuja hacia abajo. Esto mejora el rendimiento de algunas consultas. Para obtener información sobre si la consulta se puede enviar hacia abajo, consulte Pushdown en la documentación de Snowflake.

Además, algunas de las opciones disponibles en el conector Snowflake Spark pueden ser compatibles con Glue AWS. Para obtener más información sobre las opciones disponibles en el conector Snowflake Spark, consulte Configuración de las opciones de configuración del conector en la documentación de Snowflake.

Limitaciones del conector Snowflake

La conexión a Snowflake con Glue for Spark AWS está sujeta a las siguientes limitaciones.

- Este conector no admite marcadores de trabajo. Para más información acerca de los marcadores de trabajo, consulte the section called "Seguimiento de los datos procesados mediante marcadores de trabajo".
- Este conector no admite las lecturas y escrituras de Snowflake en las tablas del catálogo de datos de Glue AWS mediante los métodos `create_dynamic_frame.from_catalog` y `write_dynamic_frame.from_catalog`.
- Este conector no admite la conexión a Snowflake con credenciales distintas del usuario y la contraseña.

- Este conector no es compatible con los trabajos de streaming.
- Este conector admite consultas basadas en declaraciones `SELECT` al recuperar información (por ejemplo, con el parámetro `query`). No se admiten otros tipos de consultas (como `SHOW`, `DESC` o declaraciones DML).
- Snowflake limita el tamaño del texto de consulta (es decir, las instrucciones SQL) enviada a través de los clientes de Snowflake a 1 MB por instrucción. Para obtener más información, consulte Límites del tamaño del texto de la consulta.

Conexiones Teradata Vantage

Puede usar AWS Glue for Spark para leer y escribir en tablas de Taradata Vantage en AWS Glue 4.0 y versiones posteriores. Puede definir qué leer de Teradata con una consulta SQL. Puede conectarse a Teradata con las credenciales de nombre de usuario y contraseña almacenadas AWS Secrets Manager a través de una conexión AWS Glue.

Para obtener más información general sobre Teradata, consulte la documentación de Teradata

Configurar las conexiones de Teradata

Para conectarse a Teradata desde AWS Glue, deberá crear y almacenar las credenciales de Teradata en un AWS Secrets Manager secreto y, a continuación, asociar ese secreto a una conexión de AWS Glue Teradata. Si su instancia de Teradata está en una VPC de Amazon, también deberá proporcionar opciones de red a su conexión de AWS Glue Teradata.

Para conectarse a Teradata desde AWS Glue, es posible que necesite algunos requisitos previos:

- Si accede a su entorno de Teradata a través de Amazon VPC, configure Amazon VPC para permitir que su trabajo de AWS Glue se comunique con el entorno de Teradata. No recomendamos acceder al entorno de Teradata a través de la Internet pública.

  En Amazon VPC, identifique o cree una VPC, una subred y un grupo de seguridad que AWS Glue utilizará al ejecutar el trabajo. Además, debe asegurarse de que Amazon VPC esté configurada para permitir el tráfico de red entre su instancia de Teradata y esta ubicación. Su trabajo deberá establecer una conexión TCP con su puerto de cliente de Teradata. Para obtener más información sobre los puertos de Teradata, consulte la documentación de Teradata.

  Según el diseño de la red, la conectividad segura de la VPC puede requerir cambios en Amazon VPC y otros servicios de red. Para obtener más información sobre la conectividad de AWS, consulte las Opciones de conectividad de AWS en la documentación de Teradata.

Para configurar una conexión de AWS Glue Teradata:

1. En la configuración de Teradata, identifique o cree un usuario y una contraseña con los que se conectará AWS Glue, *teradataUser* y *teradataPassword*. Para obtener más información, consulte la Información general de seguridad de Vantage en la documentación de Teradata.
2. En AWS Secrets Manager, cree un secreto con sus credenciales de Teradata. Para crear un secreto en Secrets Manager, siga el tutorial disponible en Crear un secreto AWS Secrets Manager en la documentación AWS Secrets Manager. Después de crear el secreto, conserve el nombre secreto, *secretName*, para el siguiente paso.
   - Al seleccionar pares clave/valor, genere un par para la clave `user` con el valor *teradataUsername*.
   - Al seleccionar pares clave/valor, genere un par para la clave `password` con el valor *teradataPassword*.
3. En la consola de AWS Glue, genere una conexión mediante los pasos que se indican en the section called “Adición de una conexión de AWS Glue”. Tras crear la conexión, conserve el nombre de la conexión, *connectionName*, para el siguiente paso.
   - Al seleccionar un tipo de conexión, seleccione Teradata.
   - Al proporcionar la URL de JDBC, proporcione la URL de su instancia. También puede codificar determinados parámetros de conexión separados por comas en la URL de JDBC. La URL debe tener el siguiente formato: `jdbc:teradata://`*teradataHostname*`/`*ParameterName*`=`*ParameterValue*`,`*ParameterName*

     Los parámetros de URL admitidos incluyen:
     - `DATABASE`— nombre de la base de datos del host a la que se accede de forma predeterminada.
     - `DBS_PORT`— el puerto de la base de datos, que se utiliza cuando se ejecuta en un puerto no estándar.
   - Al seleccionar un Tipo de credencial, seleccione AWS Secrets Manager y, a continuación, establezca AWS Secret en *secretName*.
4. En las siguientes situaciones, es posible que necesite una configuración adicional:
   - Para las instancias de Teradata alojadas AWS en una VPC de Amazon

- Deberá proporcionar la información de conexión de Amazon VPC a la conexión AWS Glue que define sus credenciales de seguridad de Teradata. Al crear o actualizar la conexión, configure los VPC, Subred y los grupos de seguridad en Opciones de red.

Tras crear una conexión con AWS Glue Teradata, deberá realizar los siguientes pasos antes de llamar a su método de conexión.

- Otorgue al rol de IAM asociado al permiso de su trabajo de AWS Glue para leer *secretName*.
- En la configuración del trabajo de Glue AWS, proporcione *connectionName* como una conexión de red adicional.

Leer desde Teradata

Requisitos previos:

- Una tabla de Teradata desde la cual le gustaría leer. Necesitará el nombre de la tabla, *tableName.*
- Una conexión AWS Glue Teradata configurada para proporcionar información de autenticación. Complete los pasos de Para configurar una conexión a Teradata para y configure su información de autenticación. Necesitará el nombre de la conexión de AWS Glue, *connectionName*.

Por ejemplo:

```
teradata_read_table = glueContext.create_dynamic_frame.from_options(
    connection_type="teradata",
    connection_options={
        "connectionName": "connectionName",
        "dbtable": "tableName"
    }
)
```

También puede proporcionar una consulta SELECT SQL para filtrar los resultados devueltos a su DynamicFrame. Deberá configurar `query`.

Por ejemplo:

```
teradata_read_query = glueContext.create_dynamic_frame.from_options(
    connection_type="teradata",
```

```
    connection_options={
        "connectionName": "connectionName",
        "query": "query"
    }
)
```

Escribir en tablas de Teradata

Requisitos previos: una tabla de Teradata en la que le gustaría escribir, *tableName*. Debe crear la tabla antes de llamar al método de conexión.

Por ejemplo:

```
teradata_write = glueContext.write_dynamic_frame.from_options(
    connection_type="teradata",
    connection_options={
        "connectionName": "connectionName",
        "dbtable": "tableName"
    }
)
```

Referencia de opciones de conexión de Teradata

- `connectionName`: obligatorio. Se utiliza para lectura/escritura. El nombre de una conexión de AWS Glue Teradata configurada para proporcionar información de autenticación y red a su método de conexión.
- `dbtable` — Necesario para escribir, obligatorio para leer a menos que se proporcione `query`. Se utiliza para lectura/escritura. El nombre de una tabla con la que interactuará su método de conexión.
- `query`: se utiliza para leer. Una consulta SELECT SQL que define lo que se debe recuperar al leer desde Teradata.

Conexiones Vertica

Puede usar AWS Glue for Spark para leer y escribir en tablas de Vertica en AWS Glue 4.0 y versiones posteriores. Puede definir qué leer desde Vertica con una consulta SQL. Se conecta a Vertica con las credenciales de nombre de usuario y contraseña almacenadas en una conexión de AWS Secrets Manager a AWS Glue.

Para obtener más información sobre Vertica, consulte la documentación de Vertica.

Configurar las conexiones de Vertica

Para conectarse a Vertica desde AWS Glue, tendrá que crear y almacenar las credenciales de Vertica en un AWS Secrets Manager secreto y, a continuación, asociar ese secreto a una conexión de Vertica AWS Glue. Si su instancia de Vertica está en una VPC de Amazon, también deberá proporcionar opciones de red a su conexión de AWS Glue Vertica. Necesitará un bucket o carpeta de Amazon S3 para utilizar como almacenamiento temporario al leer y escribir en la base de datos.

Para conectarse a Vertica desde AWS Glue, necesitará algunos requisitos previos:

- Un bucket o una carpeta de Amazon S3 para utilizar como almacenamiento temporario al leer y escribir en la base de datos, al que hace referencia `tempS3Path`.

  > **Note**
  >
  > Al utilizar Vertica en las vistas previas de los datos de los trabajos de AWS Glue, puede que los archivos temporales no se eliminen automáticamente de `tempS3Path`. Para garantizar la eliminación de los archivos temporales, finalice directamente la sesión de vista previa de datos al seleccionar Finalizar sesión en el panel Vista previa de datos. Si no puede garantizar que la sesión de vista previa de datos finalice directamente, considere configurar Amazon S3 Lifecycle para eliminar los datos antiguos. Recomendamos eliminar los datos de más de 49 horas, en función del tiempo máximo de ejecución del trabajo más un margen. Para obtener más información sobre la configuración de Amazon S3 Lifecycle, consulte Administración del ciclo de vida del almacenamiento en la documentación de Amazon S3.

- Una política de IAM con los permisos adecuados para su ruta de Amazon S3 que puede asociar a su puesto de trabajo de AWS Glue.
- Si su instancia de Vertica está en una VPC de Amazon, configure Amazon VPC para permitir que su trabajo de AWS Glue se comunique con la instancia de Vertica sin que el tráfico atraviese la Internet pública.

  En Amazon VPC, identifique o cree una VPC, una subred y un grupo de seguridad que AWS Glue utilizará al ejecutar el trabajo. Además, debe asegurarse de que Amazon VPC esté configurada para permitir el tráfico de red entre su instancia de Vertica y esta ubicación. Su trabajo tendrá que establecer una conexión TCP con el puerto de cliente de Vertica (el valor predeterminado es 5433). Según el diseño de la red, esto puede requerir cambios en las reglas de los grupos de seguridad, las ACL de red, las puertas de enlace de NAT y las conexiones entre pares.

A continuación, puede proceder a configurar AWS Glue para su uso con Vertica.

Para configurar una conexión a Vertica:

1. En AWS Secrets Manager, cree un secreto con sus credenciales de Vertica, *verticaUsername* y *verticaPassword*. Para crear un secreto en Secrets Manager, siga el tutorial disponible en Crear un secreto AWS Secrets Manager en la documentación AWS Secrets Manager. Después de crear el secreto, conserve el nombre secreto, *secretName*, para el siguiente paso.

   - Al seleccionar pares clave/valor, genere un par para la clave `user` con el valor *verticaUsername*.
   - Al seleccionar pares clave/valor, genere un par para la clave `password` con el valor *verticaPassword*.

2. En la consola de AWS Glue, genere una conexión mediante los pasos que se indican en the section called “Adición de una conexión de AWS Glue”. Tras crear la conexión, conserve el nombre de la conexión, *connectionName*, para el siguiente paso.

   - Al seleccionar un tipo de conexión, seleccione Vertica.
   - Al seleccionar Vertica Host, proporcione el nombre de host de la instalación de Vertica.
   - Al seleccionar Vertica Port, proporcione el portal a través del cual está disponible su instalación de Vertica.
   - Al seleccionar un secreto AWS, proporcione un *secretName*.

3. En las siguientes situaciones, es posible que necesite una configuración adicional:

   - Para las instancias de Vertica alojadas en AWS en una VPC de Amazon
     - Proporcione la información de conexión de Amazon VPC a la conexión de AWS Glue que define sus credenciales de seguridad de Vertica. Al crear o actualizar la conexión, configure los VPC, Subred y los grupos de seguridad en Opciones de red.

Tras crear una conexión AWS Glue Vertica, deberá realizar los siguientes pasos antes de llamar a su método de conexión.

- Otorgue al rol de IAM asociado a los permisos de su trabajo de AWS Glue para *tempS3Path*.
- Otorgue al rol de IAM asociado al permiso de su trabajo de AWS Glue para leer *secretName*.

- En la configuración del trabajo de Glue AWS, proporcione *connectionName* como una conexión de red adicional.

Leer desde Vertica

Requisitos previos:

- Una tabla de Vertica desde la cual le gustaría leer. Necesitará el nombre de la base de datos Vertica, *dbName* y el nombre de la tabla, *tableName*.
- Una conexión AWS Glue Vertica configurada para proporcionar información de autenticación. Complete los pasos del procedimiento anterior, Para configurar una conexión a Vertica y configurar su información de autenticación. Necesitará el nombre de la conexión de AWS Glue, *connectionName*.
- Un bucket o una carpeta de Amazon S3 para usar como almacenamiento temporal, como se mencionó anteriormente. Necesitará el nombre *tempS3Path*. Deberá conectarse a esta ubicación mediante el protocolo `s3a`.

Por ejemplo:

```
dynamicFrame = glueContext.create_dynamic_frame.from_options(
    connection_type="vertica",
    connection_options={
        "connectionName": "connectionName",
        "staging_fs_url": "s3a://tempS3Path",
        "db": "dbName",
        "table": "tableName",
    }
)
```

También puede proporcionar una consulta de SELECT SQL para filtrar los resultados devueltos a su DynamicFrame o para acceder al conjunto de datos desde múltiples tablas.

Por ejemplo:

```
dynamicFrame = glueContext.create_dynamic_frame.from_options(
    connection_type="vertica",
    connection_options={
        "connectionName": "connectionName",
        "staging_fs_url": "s3a://tempS3Path",
```

```
        "db": "dbName",
        "query": "select * FROM tableName",
    },
)
```

Escribir en tablas de Vertica

En este ejemplo, se escribe información de un DynamicFrame existente, *dynamicFrame* en Vertica. Si la tabla ya contiene información, AWS Glue agregará los datos de su DynamicFrame.

Requisitos previos:

- El nombre de tabla actual o deseado, *tableName*, al que desearía escribir. También necesitará el nombre de la base de datos de Vertica correspondiente, *dbName.*
- Una conexión AWS Glue Vertica configurada para proporcionar información de autenticación. Complete los pasos del procedimiento anterior, Para configurar una conexión a Vertica y configurar su información de autenticación. Necesitará el nombre de la conexión de AWS Glue, *connectionName*.
- Un bucket o una carpeta de Amazon S3 para usar como almacenamiento temporal, como se mencionó anteriormente. Necesitará el nombre *tempS3Path*. Deberá conectarse a esta ubicación mediante el protocolo `s3a`.

Por ejemplo:

```
glueContext.write_dynamic_frame.from_options(
    frame=dynamicFrame,
    connection_type="vertica",
    connection_options={
        "connectionName": "connectionName",
        "staging_fs_url": "s3a://tempS3Path",
        "db": "dbName",
        "table": "tableName",
    }
)
```

Referencia de opciones de conexión de Vertica

- `connectionName`: obligatorio. Se utiliza para lectura/escritura. El nombre de una conexión de AWS Glue Vertica configurada para proporcionar información de autenticación y red a su método de conexión.

- db: obligatorio. Se utiliza para lectura/escritura. El nombre de una base de datos de Vertica con la que interactuará su método de conexión.
- dbSchema— Obligatorio si es necesario para identificar la tabla. Se utiliza para lectura/escritura. Predeterminado: public. El nombre del esquema con el que interactuará su método de conexión.
- table — Necesario para escribir, obligatorio para leer a menos que se proporcione query. Se utiliza para lectura/escritura. El nombre de una tabla con la que interactuará su método de conexión.
- query: se utiliza para leer. Una consulta SELECT SQL que define lo que se debe recuperar al leer desde Teradata.
- staging_fs_url: obligatorio. Se utiliza para lectura/escritura. Valores válidos: direcciones URL s3a. La URL de un bucket o una carpeta de Amazon S3 que se va a utilizar como almacenamiento temporal.

Valores de conexiones de tipo personalizada y AWS Marketplace

Estos incluyen los siguientes:

- "connectionType": "marketplace.athena": designa una conexión a un almacén de datos de Amazon Athena. La conexión utiliza un conector de AWS Marketplace.
- "connectionType": "marketplace.spark": designa una conexión a un almacén de datos de Apache Spark. La conexión utiliza un conector de AWS Marketplace.
- "connectionType": "marketplace.jdbc": designa una conexión a un almacén de datos de JDBC. La conexión utiliza un conector de AWS Marketplace.
- "connectionType": "custom.athena": designa una conexión a un almacén de datos de Amazon Athena. La conexión utiliza un conector personalizado que se carga a AWS Glue Studio.
- "connectionType": "custom.spark": designa una conexión a un almacén de datos de Apache Spark. La conexión utiliza un conector personalizado que se carga a AWS Glue Studio.
- "connectionType": "custom.jdbc": designa una conexión a un almacén de datos de JDBC. La conexión utiliza un conector personalizado que se carga a AWS Glue Studio.

Opciones de conexión para el tipo custom.jdbc o marketplace.jdbc

- className: cadena, obligatoria, nombre de clase de controlador.
- connectionName: cadena, obligatoria, nombre de la conexión asociada al conector.

- `url`: cadena, obligatoria, URL JDBC con marcadores de posición (`${}`) que se utilizan para construir la conexión al origen de datos. El marcador de posición `${secretKey}` se reemplaza con el secreto del mismo nombre en AWS Secrets Manager. Consulte la documentación del almacén de datos para obtener más información sobre la construcción de la URL.
- `secretId` o `user/password`: cadena, obligatoria, utilizada para recuperar credenciales de la URL.
- `dbTable` o `query`: cadena, obligatoria, la tabla o consulta SQL de la que se obtienen los datos. Puede especificar `dbTable` o `query`, pero no ambos.
- `partitionColumn`: cadena, opcional, el nombre de una columna entera que se utiliza para particionar. Esta opción solo funciona cuando está incluida con `lowerBound`, `upperBound` y `numPartitions`. Esta opción funciona de la misma manera que en el lector JDBC de Spark SQL. Para obtener más información, consulte JDBC a otras bases de datos en la Guía de Apache Spark SQL, DataFrames y conjuntos de datos.

  Los valores `lowerBound` y `upperBound` se utilizan para decidir el intervalo de partición, no para filtrar las filas de la tabla. Todas las filas de la tabla se particionan y se devuelven.

  > **Note**
  >
  > Cuando se utiliza una consulta en lugar de un nombre de tabla, debe validar que la consulta funciona con la condición de partición especificada. Por ejemplo:
  >
  > - Si el formato de consulta es `"SELECT col1 FROM table1"`, pruebe la consulta al agregar una cláusula `WHERE` al final de la consulta que utiliza la columna de partición.
  > - Si su formato de consulta es “`SELECT col1 FROM table1 WHERE col2=val`", pruebe la consulta al ampliar la cláusula `WHERE` con `AND` y una expresión que utiliza la columna de partición.

- `lowerBound`: entero, opcional, el valor mínimo de `partitionColumn` que se utiliza para decidir el intervalo de partición.
- `upperBound`: entero, opcional, el valor máximo de `partitionColumn` que se utiliza para decidir el intervalo de partición.
- `numPartitions`: entero, opcional, el número de particiones. Este valor, junto con `lowerBound` (inclusive) y `upperBound` (exclusivo), forma intervalos de partición para expresiones de la cláusula `WHERE` generadas, que se utilizan para dividir la `partitionColumn`.

> **Important**
>
> Preste atención al número de particiones, ya que demasiadas particiones pueden causar problemas en los sistemas de base de datos externos.

- `filterPredicate`: cadena, opcional, condición adicional para filtrar datos desde el origen. Por ejemplo:

  ```
  BillingCity='Mountain View'
  ```

  Cuando se utiliza una consulta en lugar de una tabla, debe validar que la consulta funciona con el `filterPredicate` especificado. Por ejemplo:

  - Si el formato de consulta es `"SELECT col1 FROM table1"`, pruebe la consulta al agregar una cláusula `WHERE` al final de la consulta que utiliza el predicado de filtrado.
  - Si su formato de consulta es `"SELECT col1 FROM table1 WHERE col2=val"`, pruebe la consulta al ampliar la cláusula `WHERE` con `AND` y una expresión que utiliza el predicado de filtrado.

- `dataTypeMapping`: diccionario, opcional, mapeo de tipos de datos personalizado, que crea un mapeo a partir de un tipo de datos JDBC a un tipo de datos de Glue. Por ejemplo, la opción `"dataTypeMapping":{"FLOAT":"STRING"}` asigna campos de datos de tipo `FLOAT` de JDBC al tipo `String` de Java al invocar el método `ResultSet.getString()` del controlador y lo utiliza para crear registros de AWS Glue. Cada controlador implementa el objeto `ResultSet`, por lo que el comportamiento es específico del controlador que se utiliza. Consulte la documentación del controlador JDBC para comprender cómo el controlador realiza las conversiones.
- Los tipos de datos de AWS Glue que se admiten actualmente son:
  - FECHA
  - STRING
  - TIMESTAMP
  - INT
  - FLOAT
  - LONG
  - BIGDECIMAL
  - BYTE

- SHORT
- DOUBLE

Los tipos de datos JDBC soportados son Java8 java.sql.types.

Las asignaciones de tipos de datos predeterminados (de JDBC a AWS Glue) son:

- DATE -> DATE
- VARCHAR -> STRING
- CHAR -> STRING
- LONGNVARCHAR -> STRING
- TIMESTAMP -> TIMESTAMP
- INTEGER -> INT
- FLOAT -> FLOAT
- REAL -> FLOAT
- BIT -> BOOLEAN
- BOOLEAN -> BOOLEAN
- BIGINT -> LONG
- DECIMAL -> BIGDECIMAL
- NUMERIC -> BIGDECIMAL
- TINYINT -> SHORT
- SMALLINT -> SHORT
- DOUBLE -> DOUBLE

Si utiliza un mapeo de tipos de datos personalizada con la opción `dataTypeMapping`, puede anular el mapeo de tipos de datos predeterminado. Sólo los tipos de datos JDBC enumerados en la opción `dataTypeMapping` se ven afectados; el mapeo predeterminado se utiliza para todos los demás tipos de datos JDBC. Puede agregar mapeos para tipos de datos JDBC adicionales si es necesario. Si un tipo de datos JDBC no está incluido en la asignación predeterminada o en una asignación personalizada, el tipo de datos se convierte al tipo de datos `STRING` de AWS Glue de forma predeterminada.

En los ejemplos de código Python siguientes, se muestra cómo leer desde bases de datos JDBC con controladores JDBC AWS Marketplace. Demuestra la lectura desde una base de datos y la escritura en una ubicación S3.

```
    import sys
    from awsglue.transforms import *
    from awsglue.utils import getResolvedOptions
    from pyspark.context import SparkContext
    from awsglue.context import GlueContext
    from awsglue.job import Job

    ## @params: [JOB_NAME]
    args = getResolvedOptions(sys.argv, ['JOB_NAME'])

    sc = SparkContext()
    glueContext = GlueContext(sc)
    spark = glueContext.spark_session
    job = Job(glueContext)
    job.init(args['JOB_NAME'], args)
    ## @type: DataSource
    ## @args: [connection_type = "marketplace.jdbc", connection_options =
     {"dataTypeMapping":{"INTEGER":"STRING"},"upperBound":"200","query":"select id,
       name, department from department where id < 200","numPartitions":"4",
       "partitionColumn":"id","lowerBound":"0","connectionName":"test-connection-
jdbc"},
        transformation_ctx = "DataSource0"]
    ## @return: DataSource0
    ## @inputs: []
    DataSource0 = glueContext.create_dynamic_frame.from_options(connection_type =
      "marketplace.jdbc", connection_options = {"dataTypeMapping":{"INTEGER":"STRING"},
      "upperBound":"200","query":"select id, name, department from department where
       id < 200","numPartitions":"4","partitionColumn":"id","lowerBound":"0",
       "connectionName":"test-connection-jdbc"}, transformation_ctx = "DataSource0")
    ## @type: ApplyMapping
    ## @args: [mappings = [("department", "string", "department", "string"), ("name",
 "string",
     "name", "string"), ("id", "int", "id", "int")], transformation_ctx =
 "Transform0"]
    ## @return: Transform0
    ## @inputs: [frame = DataSource0]
    Transform0 = ApplyMapping.apply(frame = DataSource0, mappings = [("department",
 "string",
     "department", "string"), ("name", "string", "name", "string"), ("id", "int",
 "id", "int")],
       transformation_ctx = "Transform0")
    ## @type: DataSink
    ## @args: [connection_type = "s3", format = "json", connection_options = {"path":
```

```
 "s3://<S3 path>/", "partitionKeys": []}, transformation_ctx = "DataSink0"]
## @return: DataSink0
## @inputs: [frame = Transform0]
DataSink0 = glueContext.write_dynamic_frame.from_options(frame = Transform0,
  connection_type = "s3", format = "json", connection_options = {"path":
  "s3://<S3 path>/", "partitionKeys": []}, transformation_ctx = "DataSink0")
job.commit()
```

Opciones de conexión para el tipo custom.athena o marketplace.athena

- `className`: cadena, obligatoria, nombre de clase de controlador. Cuando se utiliza el conector Athena-CloudWatch, este valor de parámetro es el prefijo del nombre de clase (por ejemplo, `"com.amazonaws.athena.connectors"`). El conector Athena-CloudWatch se compone de dos clases: un controlador de metadatos y un controlador de registros. Si proporciona el prefijo común aquí, la API carga las clases correctas basadas en ese prefijo.
- `tableName`: cadena, obligatoria, el nombre del flujo de registro de CloudWatch que se va a leer. Este fragmento de código usa el nombre de vista especial `all_log_streams`, lo que significa que el marco de datos dinámico devuelto contendrá datos de todos los flujos de registro incluidos en el grupo de registros.
- `schemaName`: cadena, obligatoria, el nombre del grupo de registro de CloudWatch que se va a leer. Por ejemplo, `/aws-glue/jobs/output`.
- `connectionName`: cadena, obligatoria, nombre de la conexión asociada al conector.

Para obtener opciones adicionales para este conector, consulte el archivo README (LÉAME) del conector de Amazon Athena CloudWatch en GitHub.

En el siguiente ejemplo de código de Python, se muestra cómo se lee desde un almacén de datos de Athena mediante un conector AWS Marketplace. Demuestra la lectura de Athena y la escritura en una ubicación S3.

```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job

## @params: [JOB_NAME]
args = getResolvedOptions(sys.argv, ['JOB_NAME'])
```

```
    sc = SparkContext()
    glueContext = GlueContext(sc)
    spark = glueContext.spark_session
    job = Job(glueContext)
    job.init(args['JOB_NAME'], args)
    ## @type: DataSource
    ## @args: [connection_type = "marketplace.athena", connection_options =
     {"tableName":"all_log_streams","schemaName":"/aws-glue/jobs/output",
      "connectionName":"test-connection-athena"}, transformation_ctx = "DataSource0"]
    ## @return: DataSource0
    ## @inputs: []
    DataSource0 = glueContext.create_dynamic_frame.from_options(connection_type =
      "marketplace.athena", connection_options = {"tableName":"all_log_streams",,
      "schemaName":"/aws-glue/jobs/output","connectionName":
      "test-connection-athena"}, transformation_ctx = "DataSource0")
    ## @type: ApplyMapping
    ## @args: [mappings = [("department", "string", "department", "string"), ("name",
"string",
     "name", "string"), ("id", "int", "id", "int")], transformation_ctx =
"Transform0"]
    ## @return: Transform0
    ## @inputs: [frame = DataSource0]
    Transform0 = ApplyMapping.apply(frame = DataSource0, mappings = [("department",
"string",
     "department", "string"), ("name", "string", "name", "string"), ("id", "int",
"id", "int")],
      transformation_ctx = "Transform0")
    ## @type: DataSink
    ## @args: [connection_type = "s3", format = "json", connection_options = {"path":
     "s3://<S3 path>/", "partitionKeys": []}, transformation_ctx = "DataSink0"]
    ## @return: DataSink0
    ## @inputs: [frame = Transform0]
    DataSink0 = glueContext.write_dynamic_frame.from_options(frame = Transform0,
      connection_type = "s3", format = "json", connection_options = {"path":
      "s3://<S3 path>/", "partitionKeys": []}, transformation_ctx = "DataSink0")
    job.commit()
```

Opciones de conexión para el tipo custom.spark o marketplace.spark

- `className`: cadena, obligatoria, nombre de clase de conector.
- `secretId`: cadena, opcional, se utiliza para recuperar las credenciales de la conexión del conector.

- `connectionName`: cadena, obligatoria, nombre de la conexión asociada al conector.
- Otras opciones dependen del almacén de datos. Por ejemplo, las opciones de configuración de OpenSearch comienzan con el prefijo `es`, tal y como se describe en la documentación Elasticsearch para Apache Hadoop. Las conexiones de Spark con Snowflake utilizan opciones tales como `sfUser` y `sfPassword`, como se describe en Uso del conector de Spark en la guía Conexión a Snowflake.

En el siguiente ejemplo de código de Python, se muestra cómo se lee desde un almacén de datos de OpenSearch mediante una conexión `marketplace.spark`.

```
   import sys
   from awsglue.transforms import *
   from awsglue.utils import getResolvedOptions
   from pyspark.context import SparkContext
   from awsglue.context import GlueContext
   from awsglue.job import Job

   ## @params: [JOB_NAME]
   args = getResolvedOptions(sys.argv, ['JOB_NAME'])

   sc = SparkContext()
   glueContext = GlueContext(sc)
   spark = glueContext.spark_session
   job = Job(glueContext)
   job.init(args['JOB_NAME'], args)
   ## @type: DataSource
   ## @args: [connection_type = "marketplace.spark", connection_options =
{"path":"test",
     "es.nodes.wan.only":"true","es.nodes":"https://<AWS endpoint>",
     "connectionName":"test-spark-es","es.port":"443"}, transformation_ctx =
"DataSource0"]
   ## @return: DataSource0
   ## @inputs: []
   DataSource0 = glueContext.create_dynamic_frame.from_options(connection_type =
     "marketplace.spark", connection_options = {"path":"test","es.nodes.wan.only":
     "true","es.nodes":"https://<AWS endpoint>","connectionName":
     "test-spark-es","es.port":"443"}, transformation_ctx = "DataSource0")
   ## @type: DataSink
   ## @args: [connection_type = "s3", format = "json", connection_options = {"path":
        "s3://<S3 path>/", "partitionKeys": []}, transformation_ctx = "DataSink0"]
   ## @return: DataSink0
```

```
    ## @inputs: [frame = DataSource0]
    DataSink0 = glueContext.write_dynamic_frame.from_options(frame = DataSource0,
       connection_type = "s3", format = "json", connection_options = {"path":
       "s3://<S3 path>/", "partitionKeys": []}, transformation_ctx = "DataSink0")
    job.commit()
```

Opciones generales

Las opciones de esta sección se proporcionan como `connection_options`, pero no se aplican específicamente a un conector.

Los siguientes parámetros se utilizan generalmente al configurar los marcadores. Se pueden aplicar a los flujos de trabajo de Amazon S3 o JDBC. Para obtener más información, consulte the section called “Uso de marcadores de trabajo”.

- `jobBookmarkKeys` — Una matriz de nombres de columna.
- `jobBookmarkKeysSortOrder` — Cadena que define cómo comparar valores en función del orden de clasificación. Valores válidos: `"asc"`, `"desc"`.
- `useS3ListImplementation` — Se utiliza para administrar el rendimiento de la memoria al publicar los contenidos de los buckets de Amazon S3. Para obtener más información, consulte Cómo optimizar la gestión de la memoria en Glue AWS.

Opciones de formato de datos para las entradas y las salidas en AWS Glue para Spark

Estas páginas ofrecen información sobre la compatibilidad de características y los parámetros de configuración para los formatos de datos compatibles con Glue AWS de Spark.. Consulte a continuación una descripción del uso y la aplicabilidad de esta información.

Compatibilidad de características en todos los formatos de datos en AWS Glue

Cada formato de datos puede admitir diferentes características de AWS Glue. Las siguientes características comunes pueden o no ser compatibles en función del tipo de formato. Consulte la documentación del formato de datos para comprender cómo aprovechar nuestras características para cumplir sus requisitos.

Leer	AWS Glue puede reconocer e interpretar este formato de datos sin recursos adicionales, como conectores.
Escritura	AWS Glue puede escribir datos en este formato sin recursos adicionales. Puede incluir bibliotecas de terceros en su trabajo y utilizar funciones estándares de Apache Spark para escribir datos, como lo haría en otros entornos de Spark. Para obtener más información sobre estas bibliotecas, consulte the section called "Bibliotecas Python".
Lectura de streamin	AWS Glue puede reconocer e interpretar este formato de datos a partir de una transmisión de mensajes de Apache Kafka, Amazon Managed Streaming para Apache Kafka o Amazon Kinesis. Esperamos que las transmisiones presenten los datos en un formato coherente, de manera que se lean en `DataFrames` .
Grupo de archivos pequeño	AWS Glue puede agrupar archivos para enviar trabajos en lotes a cada nodo cuando se realizan transformaciones de AWS Glue. Esto puede mejorar de forma significativa el rendimiento de las cargas de trabajo que implican grandes cantidades de archivos pequeños. Para obtener más información, consulte the section called "Agrupación de archivos de entrada".
Marcados de trabajo	AWS Glue puede realizar un seguimiento del progreso de las transformaciones que realizan el mismo trabajo en el mismo conjunto de datos en las ejecuciones de trabajos con marcadores de trabajo. Esto puede mejorar

el rendimiento de las cargas de trabajo que involucran conjuntos de datos en los que solo es necesario trabajar en los datos nuevos desde la última ejecución del trabajo. Para obtener más información, consulte the section called “Seguimiento de los datos procesados mediante marcadores de trabajo”.

Parámetros utilizados para interactuar con formatos de datos en AWS Glue

Ciertos tipos de conexión de AWS Glue admiten varios tipos de `format`, por lo que se requiere que especifique información sobre el formato de datos con un objeto `format_options` cuando utiliza métodos como `GlueContext.write_dynamic_frame.from_options`.

- `s3`: para obtener más información, consulte Tipos y opciones de conexión para ETL en AWS Glue: Parámetros de conexión S3. También puede ver la documentación de los métodos que facilitan este tipo de conexión: the section called “create_dynamic_frame_from_options” y the section called “write_dynamic_frame_from_options” en Python y los métodos de Scala the section called “getSourceWithFormato” y the section called “getSinkWithFormato” correspondientes.

- `kinesis`: para obtener más información, consulte Tipos y opciones de conexión para ETL en AWS Glue: Parámetros de conexión de Kinesis. También puede ver la documentación del método que facilita este tipo de conexión: the section called “create_data_frame_from_options” y el método de Scala the section called “createDataFrameFromOptions” correspondiente.
- `kafka`: para obtener más información, consulte Tipos y opciones de conexión para ETL en AWS Glue: Parámetros de conexión de Kafka. También puede ver la documentación del método que facilita este tipo de conexión: the section called “create_data_frame_from_options” y el método de Scala the section called “createDataFrameFromOptions” correspondiente.

Algunos tipos de conexión no requieren `format_options`. Por ejemplo, en uso normal, una conexión de JDBC a una base de datos relacional recupera datos en un formato de datos tabular y consistente. Por lo tanto, la lectura desde una conexión de JDBC no requeriría `format_options`.

Algunos métodos para leer y escribir datos en Glue no requieren `format_options`. Por ejemplo, el uso de `GlueContext.create_dynamic_frame.from_catalog` con rastreadores de AWS Glue. Los rastreadores determinan la forma de los datos. Cuando se utilicen rastreadores, el clasificador de AWS Glue examinará los datos para tomar decisiones inteligentes sobre cómo representar el formato

de datos. Luego, almacenará una representación de los datos en Data Catalog de AWS Glue, que se puede utilizar dentro de un script de ETL de AWS Glue para recuperar los datos con el método `GlueContext.create_dynamic_frame.from_catalog`. Los rastreadores eliminan la necesidad de especificar de forma manual la información sobre el formato de los datos.

Para trabajos que acceden a tablas que se rigen por AWS Lake Formation, AWS Glue admite la lectura y escritura de todos los formatos admitidos por las tablas que rige Lake Formation. Para obtener la lista actualizada de los formatos admitidos para tablas que se rigen por AWS Lake Formation, consulte Notas y restricciones para las tablas regidas en la Guía para desarrolladores de AWS Lake Formation.

> **Note**
>
> Para escribir Apache Parquet, ETL de AWS Glue solo admite la escritura en una tabla regida al especificar una opción para un tipo de escritor de Parquet personalizado optimizado para marcos dinámicos. Al escribir en una tabla regida con el formato de `parquet`, debe agregar la clave `useGlueParquetWriter` con un valor de `true` en los parámetros de la tabla.

Temas

- Uso del formato CSV en AWS Glue
- Uso del formato Parquet en AWS Glue
- Uso del formato XML en AWS Glue
- Uso del formato Avro en AWS Glue
- Uso del formato grokLog en AWS Glue
- Uso del formato Ion en AWS Glue
- Uso del formato JSON en AWS Glue
- Uso del formato ORC en AWS Glue
- Uso de marcos de lagos de datos con trabajos de ETL de AWS Glue
- Referencia de configuración compartida

Uso del formato CSV en AWS Glue

AWS Glue recupera datos de fuentes y escribe datos en destinos almacenados y transportados en varios formatos de datos. Si sus datos se almacenan o transportan en formato de datos CSV, este documento presenta las funciones disponibles para usar sus datos en AWS Glue.

AWS Glue admite el formato de archivo de valores separados por comas (CSV). Este formato es un formato de datos mínimo basado en filas. Los CSV a menudo no se ajustan estrictamente a un estándar, pero puede consultar RFC 4180 y RFC 7111 para obtener más información.

Puede usar AWS Glue para leer los CSV de Amazon S3 y de orígenes de streaming, o para escribir los CSV en Amazon S3. Puede leer y escribir archivos `bzip` y `gzip` que contengan archivos CSV de S3. Debe configurar el comportamiento de compresión en el Parámetros de conexión S3 en lugar de en la configuración que se describe en esta página.

En la siguiente tabla se muestran las características comunes de AWS Glue que admiten la opción de formato CSV.

Leer	Escritura	Lectura de streaming	Grupo de archivos pequeños	Marcadores de trabajo
Compatible	Soportado	Soportado	Soportado	Soportado

Ejemplo: leer archivos o carpetas CSV de S3

Requisitos previos: necesitará las rutas de S3 (`s3path`) en las carpetas o archivos CSV que desee leer.

Configuración: en las opciones de la función, especifique `format="csv"`. En sus `connection_options`, utilice la clave `paths` para especificar `s3path`. Puede configurar la forma en que el lector interactúa con S3 en `connection_options`. Para obtener más información, consulte Tipos y opciones de conexión para ETL en AWS Glue: Parámetros de conexión S3. Puede configurar el modo en que el lector interpreta los archivos CSV en su `format_options`. Para obtener más información, consulte Referencia de configuración de CSV.

El siguiente script de ETL de AWS Glue muestra el proceso de lectura de archivos o carpetas CSV desde S3.

Proporcionamos un lector de CSV personalizado con optimizaciones de rendimiento para flujos de trabajo comunes a través de la clave de configuración `optimizePerformance`. Para determinar si este lector es adecuado para su carga de trabajo, consulte the section called “Uso de un lector CSV optimizado”.

Python

Para este ejemplo, use el método create_dynamic_frame.from_options.

```
# Example: Read CSV from S3
# For show, we handle a CSV with a header row.  Set the withHeader option.
# Consider whether optimizePerformance is right for your workflow.

from pyspark.context import SparkContext
from awsglue.context import GlueContext

sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)
spark = glueContext.spark_session

dynamicFrame = glueContext.create_dynamic_frame.from_options(
    connection_type="s3",
    connection_options={"paths": ["s3://s3path"]},
    format="csv",
    format_options={
        "withHeader": True,
        # "optimizePerformance": True,
    },
)
```

También puede usar DataFrames en un script (`pyspark.sql.DataFrame`).

```
dataFrame = spark.read\
    .format("csv")\
    .option("header", "true")\
    .load("s3://s3path")
```

Scala

Para este ejemplo, use la operación getSourceWithFormat.

```
// Example: Read CSV from S3
// For show, we handle a CSV with a header row.  Set the withHeader option.
// Consider whether optimizePerformance is right for your workflow.

import com.amazonaws.services.glue.util.JsonOptions
import com.amazonaws.services.glue.{DynamicFrame, GlueContext}
import org.apache.spark.SparkContext
```

```
object GlueApp {
  def main(sysArgs: Array[String]): Unit = {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)

    val dynamicFrame = glueContext.getSourceWithFormat(
      formatOptions=JsonOptions("""{"withHeader": true}"""),
      connectionType="s3",
      format="csv",
      options=JsonOptions("""{"paths": ["s3://s3path"], "recurse": true}""")
    ).getDynamicFrame()
  }
}
```

También puede usar DataFrames en un script (`org.apache.spark.sql.DataFrame`).

```
val dataFrame = spark.read
  .option("header","true")
  .format("csv")
  .load("s3://s3path“)
```

Ejemplo: escribir archivos y carpetas CSV en S3

Requisitos previos: necesitará un DataFrame inicializado (`dataFrame`) o un DynamicFrame (`dynamicFrame`). También necesitará la ruta de salida S3 prevista, `s3path`.

Configuración: en las opciones de la función, especifique `format="csv"`. En sus `connection_options`, utilice la clave `paths` para especificar `s3path`. Puede configurar la forma en que el escritor interactúa con S3 en `connection_options`. Para obtener más información, consulte Tipos y opciones de conexión para ETL en AWS Glue: Parámetros de conexión S3. Puede configurar la forma en que la operación escribe el contenido de los archivos en `format_options`. Para obtener más información, consulte Referencia de configuración de CSV. El siguiente script de ETL de AWS Glue muestra el proceso de escritura de archivos y carpetas CSV en S3.

Python

Para este ejemplo, use el método write_dynamic_frame.from_options.

```
# Example: Write CSV to S3
```

```
# For show, customize how we write string type values.  Set quoteChar to -1 so our
 values are not quoted.

from pyspark.context import SparkContext
from awsglue.context import GlueContext

sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

glueContext.write_dynamic_frame.from_options(
    frame=dynamicFrame,
    connection_type="s3",
    connection_options={"path": "s3://s3path"},
    format="csv",
    format_options={
        "quoteChar": -1,
    },
)
```

También puede usar DataFrames en un script (`pyspark.sql.DataFrame`).

```
dataFrame.write\
    .format("csv")\
    .option("quote", None)\
    .mode("append")\
    .save("s3://s3path")
```

Scala

Para este ejemplo, use el método [getSinkWithFormat](#).

```
// Example: Write CSV to S3
// For show, customize how we write string type values. Set quoteChar to -1 so our
 values are not quoted.

import com.amazonaws.services.glue.util.JsonOptions
import com.amazonaws.services.glue.{DynamicFrame, GlueContext}
import org.apache.spark.SparkContext

object GlueApp {
  def main(sysArgs: Array[String]): Unit = {
    val spark: SparkContext = new SparkContext()
```

```
    val glueContext: GlueContext = new GlueContext(spark)

    glueContext.getSinkWithFormat(
        connectionType="s3",
        options=JsonOptions("""{"path": "s3://s3path"}"""),
        format="csv"
    ).writeDynamicFrame(dynamicFrame)
  }
}
```

También puede usar DataFrames en un script (`org.apache.spark.sql.DataFrame`).

```
dataFrame.write
    .format("csv")
    .option("quote", null)
    .mode("Append")
    .save("s3://s3path")
```

Referencia de la configuración de CVS

Puede utilizar las siguientes `format_options` donde las bibliotecas de AWS Glue especifiquen `format="csv"`:

- `separator`: especifica el carácter delimitador. El valor predeterminado es una coma, pero puede especificarse cualquier otro carácter.
  - Tipo: Texto, Valor predeterminado: `","`
- `escaper`: especifica un carácter que se usa para aplicar escape. Esta opción solo se usa cuando se leen archivos CSV, no cuando se escriben. Si se habilita, el carácter que va inmediatamente después se usa tal cual, excepto un pequeño conjunto de escapes conocidos (`\n`, `\r`, `\t` y `\0`).
  - Tipo: Texto, Valor predeterminado: ninguno
- `quoteChar`: especifica el carácter que se usa para aplicar comillas. El carácter predeterminado es una comilla doble. Establezca esta opción en `-1` para desactivar las comillas por completo.
  - Tipo: Texto, Valor predeterminado: `'"'`
- `multiLine`: especifica si un solo registro puede abarcar varias líneas. Esto puede suceder cuando un campo contiene un carácter de nueva línea entre comillas. Debe configurar esta opción en `True` si un registro abarca varias líneas. Habilitar `multiLine` podría disminuir el rendimiento ya que requiere una división de archivos más cuidadosa durante el análisis.

  - Tipo: Booleano, Valor predeterminado: `false`
- `withHeader`: especifica si la primera línea se debe tratar como un encabezado. Esta opción se puede utilizar en la clase `DynamicFrameReader`.
  - Tipo: Booleano, Valor predeterminado: `false`
- `writeHeader`: especifica si se debe escribir el encabezado en la salida. Esta opción se puede utilizar en la clase `DynamicFrameWriter`.
  - Tipo: Booleano, Valor predeterminado: `true`
- `skipFirst`: especifica si se debe omitir la primera línea de datos.
  - Tipo: Booleano, Valor predeterminado: `false`
- `optimizePerformance`: especifica si se debe utilizar el lector CSV SIMD avanzado junto con los formatos de memoria columnar basados en Apache Arrow. Solo se encuentra disponible en 3.0+ AWS Glue.
  - Tipo: Booleano, Valor predeterminado: `false`
- `strictCheckForQuoting`: al escribir archivos CSV, Glue puede agregar comillas a los valores que interpreta como cadenas. Esto se hace para evitar la ambigüedad en lo que está escrito. Para ahorrar tiempo a la hora de decidir qué escribir, Glue puede citar en determinadas situaciones en las que las citas no son necesarias. Si se habilita una verificación estricta, el cálculo será más intensivo y solo cotizará cuando sea estrictamente necesario. Solo se encuentra disponible en 3.0+ AWS Glue.
  - Tipo: Booleano, Valor predeterminado: `false`

Optimice el rendimiento de lectura con el lector CSV SIMD vectorizado

AWS Glue versión 3.0 agrega un lector de CSV optimizado que puede acelerar significativamente el rendimiento general del trabajo en comparación con los lectores de CSV basados en filas.

El lector optimizado:

- Usa las instrucciones de la CPU SIMD para leer desde el disco
- Escribe inmediatamente los registros en la memoria en un formato de columnas (Apache Arrow)
- Divide los registros en lotes

Esto ahorra tiempo de procesamiento cuando los registros se agrupan en lotes o se convierten a un formato de columnas más adelante. Algunos ejemplos son cuando se cambian esquemas o se recuperan datos por columna.

Para usar el lector optimizado, configure `"optimizePerformance"` como `true` en `format_options` o propiedad de tabla.

```
glueContext.create_dynamic_frame.from_options(
    frame = datasource1,
    connection_type = "s3",
    connection_options = {"paths": ["s3://s3path"]},
    format = "csv",
    format_options={
        "optimizePerformance": True,
        "separator": ","
        },
    transformation_ctx = "datasink2")
```

Limitaciones para el lector CSV vectorizado

Tenga en cuenta las siguientes limitaciones del lector de CSV vectorizado:

- No soporta opciones de formato `multiLine` y `escaper`. Utiliza el `escaper` predeterminado del carácter de comillas dobles `'"'`. Cuando se establecen estas opciones, AWS Glue automáticamente vuelve a utilizar el lector CSV basado en filas.
- No soporta la creación de un DynamicFrame con ChoiceType.
- No soporta la creación de un DynamicFrame con registros de error.
- No soporta la lectura de archivos CSV con caracteres multibyte, como caracteres japoneses o chinos.

Uso del formato Parquet en AWS Glue

AWS Glue recupera datos de fuentes y escribe datos en destinos almacenados y transportados en varios formatos de datos. Si sus datos se almacenan o transportan en el formato de datos Parquet, este documento presenta las funciones disponibles para usar sus datos en AWS Glue

AWS Glue admite el uso del formato Parquet. Este formato es un formato de datos basado en columnas y orientado al rendimiento. Para obtener una introducción al formato por parte de la autoridad de normalización, consulte Información general de la documentación de Apache Parquet.

Puede usar AWS Glue para leer archivos Parquet de Amazon S3 y de orígenes de streaming, como también escribir archivos Parquet en Amazon S3. Puede leer y escribir archivos `bzip` y `gzip` que contengan archivos Parquet de S3. Debe configurar el comportamiento de compresión en el Parámetros de conexión S3 en lugar de en la configuración que se describe en esta página.

En la siguiente tabla se muestran las características comunes de AWS Glue que admiten la opción de formato Parquet.

Leer	Escritura	Lectura de streaming	Grupo de archivos pequeños	Marcadores de trabajo
Compatible	Soportado	Soportado	No se admite	Compatible*

* Compatible con AWS Glue versión 1.0 y posterior

Ejemplo: leer archivos o carpetas de Parquet desde S3

Requisitos previos: necesitará las rutas de S3 (`s3path`) en las carpetas o archivos Parquet que desee leer.

Configuración: en las opciones de la función, especifique `format="parquet"`. En sus `connection_options`, utilice la clave `paths` para especificar su `s3path`.

Puede configurar la forma en que el lector interactúa con S3 en la `connection_options`. Para obtener más información, consulte Tipos y opciones de conexión para ETL en AWS Glue: Parámetros de conexión S3.

Puede configurar la forma en que el lector interpreta los archivos Parquet en sus `format_options`. Para obtener más información, consulte Referencia de configuración de Parquet.

El siguiente script de ETL de AWS Glue muestra el proceso de lectura de archivos o carpetas Parquet desde S3:

Python

Para este ejemplo, use el método create_dynamic_frame.from_options.

```
# Example: Read Parquet from S3
```

```
from pyspark.context import SparkContext
from awsglue.context import GlueContext

sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)
spark = glueContext.spark_session

dynamicFrame = glueContext.create_dynamic_frame.from_options(
    connection_type = "s3",
    connection_options = {"paths": ["s3://s3path/"]},
    format = "parquet"
)
```

También puede usar DataFrames en un script (`pyspark.sql.DataFrame`).

```
dataFrame = spark.read.parquet("s3://s3path/")
```

Scala

Para este ejemplo, use el método getSourceWithFormat.

```
// Example: Read Parquet from S3

import com.amazonaws.services.glue.util.JsonOptions
import com.amazonaws.services.glue.{DynamicFrame, GlueContext}
import org.apache.spark.SparkContext

object GlueApp {
  def main(sysArgs: Array[String]): Unit = {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)

    val dynamicFrame = glueContext.getSourceWithFormat(
      connectionType="s3",
      format="parquet",
      options=JsonOptions("""{"paths": ["s3://s3path"]}""")
    ).getDynamicFrame()
  }
}
```

También puede usar DataFrames en un script (`org.apache.spark.sql.DataFrame`).

```
spark.read.parquet("s3://s3path/")
```

Ejemplo: escribir archivos y carpetas de Parquet en S3

Requisitos previos: necesitará un DataFrame inicializado (`dataFrame`) o DynamicFrame (`dynamicFrame`). También necesitará la ruta de salida S3 prevista, `s3path`.

Configuración: en las opciones de la función, especifique `format="parquet"`. En sus `connection_options`, utilice la clave `paths` para especificar `s3path`.

Puede modificar aún más la forma en que el escritor interactúa con S3 en las `connection_options`. Para obtener más información, consulte Tipos y opciones de conexión para ETL en AWS Glue: Parámetros de conexión S3. Puede configurar la forma en que la operación escribe el contenido de los archivos en `format_options`. Para obtener más información, consulte Referencia de configuración de Parquet.

El siguiente script de ETL de AWS Glue muestra el proceso de escritura de archivos y carpetas Parquet en S3.

Proporcionamos un escritor de Parquet personalizado con optimizaciones de rendimiento para DynamicFrames, a través de la clave de la configuración `useGlueParquetWriter`. Para determinar si este escritor es adecuado para su carga de trabajo, consulte Escritor de Glue Parquet.

Python

Para este ejemplo, use el método write_dynamic_frame.from_options.

```
# Example: Write Parquet to S3
# Consider whether useGlueParquetWriter is right for your workflow.

from pyspark.context import SparkContext
from awsglue.context import GlueContext

sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

glueContext.write_dynamic_frame.from_options(
    frame=dynamicFrame,
    connection_type="s3",
    format="parquet",
```

```
    connection_options={
        "path": "s3://s3path",
    },
    format_options={
        # "useGlueParquetWriter": True,
    },
)
```

También puede usar DataFrames en un script (`pyspark.sql.DataFrame`).

```
df.write.parquet("s3://s3path/")
```

Scala

Para este ejemplo, use el método getSinkWithFormat.

```
// Example: Write Parquet to S3
// Consider whether useGlueParquetWriter is right for your workflow.

import com.amazonaws.services.glue.util.JsonOptions
import com.amazonaws.services.glue.{DynamicFrame, GlueContext}
import org.apache.spark.SparkContext

object GlueApp {
  def main(sysArgs: Array[String]): Unit = {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)

    glueContext.getSinkWithFormat(
        connectionType="s3",
        options=JsonOptions("""{"path": "s3://s3path"}"""),
        format="parquet"
    ).writeDynamicFrame(dynamicFrame)
  }
}
```

También puede usar DataFrames en un script (`org.apache.spark.sql.DataFrame`).

```
df.write.parquet("s3://s3path/")
```

Referencia de configuración de Parquet

Puede utilizar las siguientes `format_options` donde las bibliotecas de AWS Glue especifiquen `format="parquet"`:

- `useGlueParquetWriter`: especifica el uso de un escritor Parquet personalizado que tiene optimizaciones de rendimiento para flujos de trabajo de DynamicFrame. Para obtener más información sobre el uso, consulte Escritor de Glue Parquet.
  - Tipo: Booleano, Valor predeterminado:`false`
- `compression`: especifica el códec de compresión utilizado. Los valores son totalmente compatibles con `org.apache.parquet.hadoop.metadata.CompressionCodecName`.
  - Tipo: Texto enumerado, Valor predeterminado: `"snappy"`
  - Valores: `"uncompressed"`, `"snappy"`, `"gzip"` y `"lzo"`
- `blockSize`: especifica el tamaño en bytes de un grupo de filas que se están almacenando en el búfer. Se usa para ajustar el rendimiento. El tamaño debe dividirse exactamente en un número de megabytes.
  - Tipo: Numérico, Valor predeterminado:`134217728`
  - El valor predeterminado es igual a 128 MB.
- `pageSize`: especifica el tamaño en bytes de una página. Se usa para ajustar el rendimiento. Una página es la unidad más pequeña que debe leerse por completo para obtener acceso a un único registro.
  - Tipo: Numérico, Valor predeterminado:`1048576`
  - El valor predeterminado es igual a 1 MB.

> **Note**
>
> Además, se pueden transferir a este formato las opciones que acepte el código SparkSQL subyacente mediante el parámetro de mapa `connection_options`. Por ejemplo, se puede establecer una configuración de Spark como mergeSchema para que el lector de Spark de AWS Glue fusione el esquema para todos los archivos.

Optimice el rendimiento de escritura con el escritor de AWS Glue Parquet

> **Note**
>
> Históricamente, se ha accedido al escritor de AWS Glue Parquet mediante el tipo de formato `glueparquet`. Ya no se aboga por este patrón de acceso. En su lugar, utilice el tipo `parquet` con `useGlueParquetWriter` habilitado.

El escritor AWS Glue Parquet tiene mejoras de rendimiento que permiten escribir archivos Parquet más rápidamente. El escritor tradicional calcula un esquema antes de escribir. El formato Parquet no almacena el esquema de forma que se pueda recuperar rápidamente, por lo que puede llevar algún tiempo. Con el escritor de AWS Glue Parquet, no se requiere un esquema precalculado. A medida que llegan los datos, el escritor calcula y modifica el esquema e forma dinámica.

Cuando especifique `useGlueParquetWriter`, tenga en cuenta las siguientes limitaciones:

- El escritor solo admite la evolución de los esquemas (como agregar o eliminar columnas) pero no permite cambiar los tipos de columnas, como sucede con `ResolveChoice`.
- El escritor no puede almacenar un DataFrame vacío, por ejemplo, para escribir un archivo solo de esquema. Al realizar la integración con el Catálogo de datos de AWS Glue mediante la configuración de `enableUpdateCatalog=True`, si se intenta escribir un DataFrame vacío, no se actualizará el Catálogo de datos. Esto creará una tabla sin esquema en el Catálogo de datos.

Si su transformación no requiere estas limitaciones, activar el escritor AWS Glue Parquet debería aumentar el rendimiento.

Uso del formato XML en AWS Glue

AWS Glue recupera datos de fuentes y escribe datos en destinos almacenados y transportados en varios formatos de datos. Si sus datos se almacenan o transportan en formato de datos XML, este documento presenta las funciones disponibles para usar sus datos en AWS Glue.

AWS Glue admite el uso del formato XML. Este formato representa estructuras de datos altamente configurables y rígidamente definidas que no están basadas en filas o columnas. XML es altamente estandarizado. Para obtener una introducción al formato de la autoridad de normalización, consulte Elementos esenciales de XML.

Puede usar AWS Glue para leer archivos XML desde Amazon S3, así como archivos `bzip` y `gzip` que contengan archivos XML. Debe configurar el comportamiento de compresión en el Parámetros de conexión S3 en lugar de en la configuración que se describe en esta página.

En la siguiente tabla se muestran las características comunes de AWS Glue que admiten la opción de formato XML.

Leer	Escritura	Lectura de streaming	Grupo de archivos pequeños	Marcadores de trabajo
Compatible	No se admite	No se admite	Soportado	Soportado

Ejemplo: leer XML desde S3

El lector de XML toma un nombre de etiqueta XML. Examina los elementos con esa etiqueta dentro de su entrada para inferir un esquema y rellena un DynamicFrame con los valores correspondientes. La funcionalidad XML de AWS Glue se comporta de forma similar al Origen de datos XML para Apache Spark. Es posible que pueda obtener información sobre el comportamiento básico comparando este lector con la documentación de ese proyecto.

Requisitos previos: necesitará las rutas de S3 (`s3path`) en las carpetas o archivos XML que desee leer, y cierta información sobre su archivo XML. También necesitará la etiqueta del elemento XML que desea leer, `xmlTag`.

Configuración: en las opciones de la función, especifique `format="xml"`. En sus `connection_options`, utilice la clave `paths` para especificar `s3path`. Puede configurar aún más la forma en que el lector interactúa con S3 en la `connection_options`. Para obtener más información, consulte Tipos y opciones de conexión para ETL en AWS Glue: Parámetros de conexión S3. En sus `format_options`, utilice la clave `rowTag` para especificar `xmlTag`. Puede configurar aún más la forma en que el lector interpreta los archivos XML en sus `format_options`. Para obtener más información, consulte Referencia de configuración de XML.

El siguiente script de ETL de AWS Glue muestra el proceso de lectura de archivos o carpetas XML desde S3.

Python

Para este ejemplo, use el método create_dynamic_frame.from_options.

```
# Example: Read XML from S3
# Set the rowTag option to configure the reader.

from awsglue.context import GlueContext
from pyspark.context import SparkContext

sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

dynamicFrame = glueContext.create_dynamic_frame.from_options(
    connection_type="s3",
    connection_options={"paths": ["s3://s3path"]},
    format="xml",
    format_options={"rowTag": "xmlTag"},
)
```

También puede usar DataFrames en un script (`pyspark.sql.DataFrame`).

```
dataFrame = spark.read\
    .format("xml")\
    .option("rowTag", "xmlTag")\
    .load("s3://s3path")
```

Scala

Para este ejemplo, use la operación getSourceWithFormat.

```
// Example: Read XML from S3
// Set the rowTag option to configure the reader.

import com.amazonaws.services.glue.util.JsonOptions
import com.amazonaws.services.glue.GlueContext
import org.apache.spark.sql.SparkSession

val glueContext = new GlueContext(SparkContext.getOrCreate())
val sparkSession: SparkSession = glueContext.getSparkSession

object GlueApp {
  def main(sysArgs: Array[String]): Unit = {
    val dynamicFrame = glueContext.getSourceWithFormat(
      formatOptions=JsonOptions("""{"rowTag": "xmlTag"}"""),
      connectionType="s3",
```

```
      format="xml",
      options=JsonOptions("""{"paths": ["s3://s3path"], "recurse": true}""")
    ).getDynamicFrame()
}
```

También puede usar DataFrames en un script (`org.apache.spark.sql.DataFrame`).

```
val dataFrame = spark.read
  .option("rowTag", "xmlTag")
  .format("xml")
  .load("s3://s3path“)
```

Referencia de configuración de XML

Puede utilizar las siguientes `format_options` donde las bibliotecas de AWS Glue especifiquen `format="xml"`:

- `rowTag`: especifica la etiqueta XML en el archivo que se tratará como una fila. Las etiquetas de fila no pueden autocerrarse.
  - Tipo: Texto, Obligatorio
- `encoding`: especifica la codificación de caracteres. Puede ser el nombre o alias de un Charset compatible con nuestro entorno de tiempo de ejecución. No ofrecemos garantías específicas en cuanto a la compatibilidad con la codificación, pero las codificaciones principales deberían funcionar.
  - Tipo: Texto, Valor predeterminado: `"UTF-8"`
- `excludeAttribute`: especifica si desea excluir o no atributos en elementos.
  - Tipo: Booleano, Valor predeterminado: `false`
- `treatEmptyValuesAsNulls`: especifica si los espacios en blanco se tratan como un valor nulo.
  - Tipo: Booleano, Valor predeterminado: `false`
- `attributePrefix`: un prefijo de atributos para diferenciarlos del texto de elementos secundarios. Este prefijo se utiliza para los nombres de campo.
  - Tipo: Texto, Valor predeterminado: `"_"`
- `valueTag`: la etiqueta que se utiliza para un valor cuando hay atributos en el elemento que no tienen elementos secundarios.
  - Tipo: Texto, Valor predeterminado: `"_VALUE"`

- `ignoreSurroundingSpaces`: especifica si se deben ignorar los espacios en blanco alrededor de los valores.
  - Tipo: Booleano, Valor predeterminado: `false`
- `withSchema`: contiene el esquema esperado, en situaciones en las que se desea anular el esquema inferido. Si no utiliza esta opción, AWS Glue infiere el esquema a partir de los datos XML.
  - Tipo: Texto, Valor predeterminado: no aplicable
  - El valor debe ser un objeto JSON que represente un `StructType`.

Especifique manualmente el esquema XML

Ejemplo del esquema XML manual

Este es un ejemplo de uso de la opción del formato `withSchema` para especificar el esquema para los datos XML.

```
from awsglue.gluetypes import *

schema = StructType([
  Field("id", IntegerType()),
  Field("name", StringType()),
  Field("nested", StructType([
    Field("x", IntegerType()),
    Field("y", StringType()),
    Field("z", ChoiceType([IntegerType(), StringType()]))
  ]))
])

datasource0 = create_dynamic_frame_from_options(
    connection_type,
    connection_options={"paths": ["s3://xml_bucket/someprefix"]},
    format="xml",
    format_options={"withSchema": json.dumps(schema.jsonValue())},
    transformation_ctx = ""
)
```

Uso del formato Avro en AWS Glue

AWS Glue recupera datos de fuentes y escribe datos en destinos almacenados y transportados en varios formatos de datos. Si los datos se almacenan o se transportan en formato de datos Avro, este documento presenta las características disponibles para utilizar los datos en AWS Glue.

AWS Glue admite el uso del formato Avro. Este es un formato de datos basado en filas y orientado al rendimiento. Para obtener una introducción al formato por parte de la autoridad de normalización, consulte Documentación de Apache Avro 1.8.2.

Puede utilizar AWS Glue para leer archivos Avro de Amazon S3 y de orígenes de streaming, como también para escribir archivos Avro en Amazon S3. Puede leer y escribir archivos `bzip2` y `gzip` que contengan archivos Avro de S3. Además, puede escribir `deflate`, `snappy` y `xz` archivos que contengan archivos Avro. Debe configurar el comportamiento de compresión en el Parámetros de conexión S3 en lugar de en la configuración que se describe en esta página.

En la siguiente tabla se muestran las operaciones comunes de AWS Glue que admiten la opción de formato Avro.

Leer	Escritura	Lectura de streaming	Grupo de archivos pequeños	Marcadores de trabajo
Compatible	Soportado	Compatible*	No se admite	Compatible

*Compatible con restricciones. Para obtener más información, consulte the section called “Notas y restricciones para orígenes de streaming de Avro”.

Ejemplo: leer archivos o carpetas Avro de S3

Requisitos previos: necesitará las rutas de S3 (`s3path`) de las carpetas o los archivos Avro que desee leer.

Configuración: en las opciones de la función, especifique `format="avro"`. En sus `connection_options`, utilice la clave `paths` para especificar `s3path`. Puede configurar la forma en que el lector interactúa con S3 en la `connection_options`. Para obtener más información, consulte Opciones de formato de datos para entradas y salidas de ETL en AWS Glue: the section called “Parámetros de conexión S3”. Puede configurar la manera en que el lector interpreta los

archivos Avro en sus `format_options`. Para obtener más información, consulte Referencia de configuración de Avro.

El siguiente script de ETL de AWS Glue muestra el proceso de lectura de archivos o carpetas Avro de S3:

Python

Para este ejemplo, use el método create_dynamic_frame.from_options.

```
from pyspark.context import SparkContext
from awsglue.context import GlueContext

sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

dynamicFrame = glueContext.create_dynamic_frame.from_options(
    connection_type="s3",
    connection_options={"paths": ["s3://s3path"]},
    format="avro"
)
```

Scala

Para este ejemplo, use la operación getSourceWithFormat.

```
import com.amazonaws.services.glue.util.JsonOptions
import com.amazonaws.services.glue.GlueContext
import org.apache.spark.sql.SparkContext

object GlueApp {
  def main(sysArgs: Array[String]): Unit = {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)

    val dynamicFrame = glueContext.getSourceWithFormat(
      connectionType="s3",
      format="avro",
      options=JsonOptions("""{"paths": ["s3://s3path"]}""")
    ).getDynamicFrame()
  }
```

Ejemplo: escribir archivos y carpetas Avro en S3

Requisitos previos: necesitará un DataFrame inicializado (`dataFrame`) o DynamicFrame (`dynamicFrame`). También necesitará la ruta de salida S3 prevista, `s3path`.

Configuración: en las opciones de la función, especifique `format="avro"`. En sus `connection_options`, utilice la clave `paths` para especificar su `s3path`. Puede modificar aún más la forma en que el escritor interactúa con S3 en las `connection_options`. Para obtener más información, consulte Opciones de formato de datos para entradas y salidas de ETL en AWS Glue: the section called “Parámetros de conexión S3”. Puede alterar la manera en que el escritor interpreta los archivos Avro en sus `format_options`. Para obtener más información, consulte Referencia de configuración de Avro.

El siguiente script de ETL de AWS Glue muestra el proceso de escritura de archivos o carpetas Avro en S3.

Python

Para este ejemplo, use el método write_dynamic_frame.from_options.

```
from pyspark.context import SparkContext
from awsglue.context import GlueContext

sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

glueContext.write_dynamic_frame.from_options(
    frame=dynamicFrame,
    connection_type="s3",
    format="avro",
    connection_options={
        "path": "s3://s3path"
    }
)
```

Scala

Para este ejemplo, use el método getSinkWithFormat.

```
import com.amazonaws.services.glue.util.JsonOptions
import com.amazonaws.services.glue.{DynamicFrame, GlueContext}
import org.apache.spark.SparkContext
```

```
object GlueApp {
  def main(sysArgs: Array[String]): Unit = {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)

    glueContext.getSinkWithFormat(
      connectionType="s3",
      options=JsonOptions("""{"path": "s3://s3path"}"""),
      format="avro"
    ).writeDynamicFrame(dynamicFrame)
  }
}
```

Referencia de configuración de Avro

Puede utilizar los siguientes valores `format_options` donde las bibliotecas de AWS Glue especifiquen `format="avro"`:

- `version`: especifica la versión del formato lector/escritor Apache Avro que se va a soportar. El valor predeterminado es 1.7. Puede especificar `format_options={"version": “1.8”}` para habilitar el tipo de lectura y escritura lógico de Avro. Para obtener más información, consulte la especificación de Apache Avro 1.7.7 y la especificación de Apache Avro 1.8.2.

  El conector de Apache Avro 1.8 admite las siguientes conversiones de tipo lógico:

Para el lector: en esta tabla, se muestra la conversión entre el tipo de datos de Avro (tipo lógico y tipo primitivo de Avro) y el tipo de datos `DynamicFrame` de AWS Glue para las versiones 1.7 y 1.8 del lector de Avro.

Tipo de datos de Avro: Tipo lógico	Tipo de datos de Avro: Tipo primitivo de Avro	Tipo de datos DynamicFrame de Glue: Lector 1.7 de Avro	Tipo de datos DynamicFrame de Glue: Lector 1.8 de Avro
Decimal	bytes	BINARY	Decimal
Decimal	fijo	BINARY	Decimal

Tipo de datos de Avro: Tipo lógico	Tipo de datos de Avro: Tipo primitivo de Avro	Tipo de datos DynamicFrame de Glue: Lector 1.7 de Avro	Tipo de datos DynamicFrame de Glue: Lector 1.8 de Avro
Date	int	INT	Date
Tiempo (milisegundos)	int	INT	INT
Tiempo (microsegundos)	long	LONG	LONG
Marca de tiempo (milisegundos)	long	LONG	Timestamp
Marca de tiempo (microsegundos)	long	LONG	LONG
Duración (no es un tipo lógico)	fijo de 12	BINARY	BINARY

Para el escritor: en esta tabla, se muestra la conversión entre el tipo de datos `DynamicFrame` de AWS Glue y el tipo de datos de Avro para las versiones 1.7 y 1.8 del escritor de Avro.

Tipo de datos **`DynamicFrame`** de AWS Glue	Tipo de datos de Avro: Escritor 1.7 de Avro	Tipo de datos de Avro: Escritor 1.8 de Avro
Decimal	Cadena	decimal
Date	Cadena	date
Timestamp	Cadena	marca temporal (microsegundos)

Compatibilidad entre Avro y Spark DataFrame

Para utilizar Avro desde la API de Spark DataFrame, debe instalar el complemento Spark Avro para la versión de Spark correspondiente. La versión de Spark disponible en su trabajo está determinada por la versión de AWS Glue. Para obtener más información acerca de las versiones de Spark, consulte the section called “Versiones de AWS Glue”. Apache mantiene este complemento; nosotros no ofrecemos garantías específicas de compatibilidad.

En AWS Glue 2.0 se utiliza la versión 2.4.3 del complemento Spark Avro. Puede encontrar este JAR en Maven Central, consulte org.apache.spark:spark-avro_2.12:2.4.3.

En AWS Glue 3.0 se utiliza la versión 3.1.1 del complemento Spark Avro. Puede encontrar este JAR en Maven Central, consulte org.apache.spark:spark-avro_2.12:3.1.1.

Para incluir archivos JAR adicionales en un trabajo de ETL de AWS Glue, utilice el parámetro de trabajo `--extra-jars`. Para obtener más información acerca de la configuración de parámetros de trabajos, consulte the section called “Parámetros del flujo de trabajo”. También puede configurar este parámetro en la AWS Management Console.

Uso del formato grokLog en AWS Glue

AWS Glue recupera datos de fuentes y escribe datos en destinos almacenados y transportados en varios formatos de datos. Si los datos se almacenan o se transportan en texto sin formato y en un formato poco estructurado, este documento presenta las características disponibles para utilizar los datos en AWS Glue a través de patrones de Grok.

AWS Glue admite el uso de patrones de Grok. Los patrones de Grok son similares a los grupos de captura de expresiones regulares. Reconocen patrones de secuencias de caracteres en un archivo de texto sin formato y les dan un tipo y un propósito. En AWS Glue, su objetivo principal es leer registros. Para obtener una introducción a Grok por parte de los autores, consulte Referencia de Logstash: complemento de filtro de Grok.

Leer	Escritura	Lectura de streaming	Grupo de archivos pequeños	Marcadores de trabajo
Compatible	No aplicable	Soportado	Soportado	No se admite

Referencia de configuración de grokLog

Puede utilizar los siguientes valores `format_options` con `format="grokLog"`:

- `logFormat`: especifica el patrón de Grok que coincide con el formato del registro.
- `customPatterns`: especifica los patrones de Grok adicionales que se utilizan aquí.
- `MISSING`: especifica la señal que se utilizará en la identificación de los valores que faltan. El valor predeterminado es `'-'`.
- `LineCount`: especifica el número de líneas en cada registro. El valor predeterminado es `'1'` y actualmente solo se admiten los registros de una sola línea.
- `StrictMode`: un valor booleano que especifica si el modo estricto está habilitado. En modo estricto, el lector no efectúa la conversión o recuperación de tipo automática. El valor predeterminado es `"false"`.

Uso del formato Ion en AWS Glue

AWS Glue recupera datos de fuentes y escribe datos en destinos almacenados y transportados en varios formatos de datos. Si los datos se almacenan o se transportan en formato de datos Ion, este documento presenta las características disponibles para utilizar los datos en AWS Glue.

AWS Glue admite el uso del formato Ion. Este formato representa estructuras de datos (que no están basadas en filas o columnas) en representaciones binarias y de texto sin formato intercambiables. Para obtener una introducción al formato por parte de los autores, consulte Amazon Ion. (Para obtener más información, consulte Especificación de Amazon Ion).

Puede utilizar AWS Glue para leer archivos Ion de Amazon S3. Puede leer y escribir archivos `bzip` y `gzip` que contengan archivos Ion de S3. Debe configurar el comportamiento de compresión en el Parámetros de conexión S3 en lugar de en la configuración que se describe en esta página.

En la siguiente tabla se muestran las operaciones comunes de AWS Glue que admiten la opción de formato Ion.

Leer	Escritura	Lectura de streaming	Grupo de archivos pequeños	Marcadores de trabajo
Compatible	No se admite	No se admite	Compatible	No se admite

Ejemplo: leer archivos o carpetas Ion de S3

Requisitos previos: necesitará las rutas de S3 (`s3path`) de las carpetas o los archivos Ion que desee leer.

Configuración: en las opciones de la función, especifique `format="json"`. En sus `connection_options`, utilice la clave `paths` para especificar su `s3path`. Puede configurar la forma en que el lector interactúa con S3 en la `connection_options`. Para obtener más información, consulte Tipos y opciones de conexión para ETL en AWS Glue: the section called "Parámetros de conexión S3".

El siguiente script de ETL de AWS Glue muestra el proceso de lectura de archivos o carpetas Ion de S3:

Python

Para este ejemplo, use el método create_dynamic_frame.from_options.

```
# Example: Read ION from S3

from pyspark.context import SparkContext
from awsglue.context import GlueContext

sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

dynamicFrame = glueContext.create_dynamic_frame.from_options(
    connection_type="s3",
    connection_options={"paths": ["s3://s3path"]},
    format="ion"
)
```

Scala

Para este ejemplo, use la operación getSourceWithFormat.

```
// Example: Read ION from S3

import com.amazonaws.services.glue.util.JsonOptions
import com.amazonaws.services.glue.GlueContext
import org.apache.spark.SparkContext
```

```
object GlueApp {
  def main(sysArgs: Array[String]): Unit = {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)

    val dynamicFrame = glueContext.getSourceWithFormat(
      connectionType="s3",
      format="ion",
      options=JsonOptions("""{"paths": ["s3://s3path"], "recurse": true}""")
    ).getDynamicFrame()
  }
}
```

Referencia de configuración de Ion

No hay valores `format_options` para `format="ion"`.

Uso del formato JSON en AWS Glue

AWS Glue recupera los datos de las fuentes y los escribe en los destinos almacenados y transportados en varios formatos de datos. Si tus datos se almacenan o transportan en el formato de datos JSON, este documento te presenta las funciones disponibles para usar tus datos en AWS Glue.

AWS Glue admite el uso del formato JSON. Este formato representa estructuras de datos con forma consistente, pero con contenido flexible, que no están basadas en filas o columnas. JSON está definido por estándares paralelos emitidos por varias autoridades, una de las cuales es ECMA-404. Para obtener una introducción al formato por parte de un origen al que normalmente se hace referencia, consulte Introducción a JSON.

Puede usar AWS Glue para leer archivos JSON de Amazon S3, así como `bzip` archivos JSON `gzip` comprimidos. Debe configurar el comportamiento de compresión en el Parámetros de conexión S3 en lugar de en la configuración que se describe en esta página.

Leer	Escritura	Lectura de streaming	Grupo de archivos pequeños	Marcadores de trabajo	
Compatible	Soportado	Soportado	Soportado	Soportado	

Ejemplo: leer archivos o carpetas JSON de S3

Requisitos previos: necesitará las rutas de S3 (`s3path`) de las carpetas o los archivos JSON que desee leer.

Configuración: en las opciones de la función, especifique `format="json"`. En sus `connection_options`, utilice la clave `paths` para especificar su `s3path`. Puede modificar aún más la manera en que la operación de lectura atravesará S3 en las opciones de conexión; consulte the section called “Parámetros de conexión S3” para obtener más detalles. Puede configurar la manera en que el lector interpreta los archivos JSON en sus `format_options`. Para obtener más información, consulte Referencia de configuración de JSON.

El siguiente script ETL de AWS Glue muestra el proceso de lectura de archivos o carpetas JSON de S3:

Python

Para este ejemplo, use el método create_dynamic_frame.from_options.

```
# Example: Read JSON from S3
# For show, we handle a nested JSON file that we can limit with the JsonPath
 parameter
# For show, we also handle a JSON where a single entry spans multiple lines
# Consider whether optimizePerformance is right for your workflow.

from pyspark.context import SparkContext
from awsglue.context import GlueContext

sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)
spark = glueContext.spark_session

dynamicFrame = glueContext.create_dynamic_frame.from_options(
    connection_type="s3",
    connection_options={"paths": ["s3://s3path"]},
    format="json",
    format_options={
        "jsonPath": "$.id",
        "multiline": True,
        # "optimizePerformance": True, -> not compatible with jsonPath, multiline
    }
)
```

También se puede usar DataFrames en un script (`pyspark.sql.DataFrame`).

```
dataFrame = spark.read\
    .option("multiLine", "true")\
    .json("s3://s3path")
```

Scala

Para este ejemplo, utilice la operación de getSourceWithformato.

```
// Example: Read JSON from S3
// For show, we handle a nested JSON file that we can limit with the JsonPath
 parameter
// For show, we also handle a JSON where a single entry spans multiple lines
// Consider whether optimizePerformance is right for your workflow.

import com.amazonaws.services.glue.util.JsonOptions
import com.amazonaws.services.glue.{DynamicFrame, GlueContext}
import org.apache.spark.SparkContext

object GlueApp {
  def main(sysArgs: Array[String]): Unit = {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)

    val dynamicFrame = glueContext.getSourceWithFormat(
      formatOptions=JsonOptions("""{"jsonPath": "$.id", "multiline": true,
 "optimizePerformance":false}"""),
      connectionType="s3",
      format="json",
      options=JsonOptions("""{"paths": ["s3://s3path"], "recurse": true}""")
    ).getDynamicFrame()
  }
}
```

También se puede utilizar DataFrames en un script (`pyspark.sql.DataFrame`).

```
val dataFrame = spark.read
    .option("multiLine", "true")
    .json("s3://s3path")
```

Ejemplo: escribir archivos y carpetas JSON en S3

Requisitos previos: Necesitará un DataFrame (`dataFrame`) o DynamicFrame (`dynamicFrame`) inicializado. También necesitará la ruta de salida S3 prevista, `s3path`.

Configuración: en las opciones de la función, especifique `format="json"`. En sus `connection_options`, utilice la clave `paths` para especificar `s3path`. Puede modificar aún más la forma en que el escritor interactúa con S3 en las `connection_options`. Para obtener más información, consulte Opciones de formato de datos para entradas y salidas ETL en AWS Glue:the section called “Parámetros de conexión S3”. Puede configurar la forma en que el escritor interpreta los archivos JSON en sus `format_options`. Para obtener más información, consulte Referencia de configuración de JSON.

El siguiente script ETL de AWS Glue muestra el proceso de escritura de archivos o carpetas JSON desde S3:

Python

Para este ejemplo, use el método write_dynamic_frame.from_options.

```
# Example: Write JSON to S3

from pyspark.context import SparkContext
from awsglue.context import GlueContext

sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

glueContext.write_dynamic_frame.from_options(
    frame=dynamicFrame,
    connection_type="s3",
    connection_options={"path": "s3://s3path"},
    format="json"
)
```

También se puede usar DataFrames en un script (`pyspark.sql.DataFrame`).

```
df.write.json("s3://s3path/")
```

Scala

Para este ejemplo, utilice el método getSinkWithFormat.

```
// Example: Write JSON to S3

import com.amazonaws.services.glue.util.JsonOptions
import com.amazonaws.services.glue.{DynamicFrame, GlueContext}
import org.apache.spark.SparkContext

object GlueApp {
  def main(sysArgs: Array[String]): Unit = {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)

    glueContext.getSinkWithFormat(
        connectionType="s3",
        options=JsonOptions("""{"path": "s3://s3path"}"""),
        format="json"
    ).writeDynamicFrame(dynamicFrame)
  }
}
```

También se puede utilizar DataFrames en un script (`pyspark.sql.DataFrame`).

```
df.write.json("s3://s3path")
```

Referencia de configuración de JSON

Puede utilizar los siguientes valores `format_options` con `format="json"`:

- `jsonPath`— JsonPathExpresión que identifica un objeto que se va a leer en los registros. Esto resulta particularmente útil cuando un archivo contiene registros anidados en una matriz exterior. Por ejemplo, la siguiente JsonPath expresión apunta al `id` campo de un objeto JSON.

  ```
  format="json", format_options={"jsonPath": "$.id"}
  ```

- `multiLine`: un valor booleano que especifica si un solo registro puede abarcar varias líneas. Esto puede suceder cuando un campo contiene un carácter de nueva línea entre comillas. Debe configurar esta opción en `"true"` si un registro abarca varias líneas. El valor predeterminado es `"false"`, que permite una división de archivo más dinámica durante el análisis.
- `optimizePerformance`: un valor booleano que especifica si se debe utilizar el lector SIMD JSON avanzado junto con los formatos de memoria en columnas basados en Apache Arrow.

Solo se encuentra disponible en AWS Glue 3.0. No compatible con `multiLine` ni `jsonPath`. Al proporcionar cualquiera de esas opciones, AWS Glue tendrá que recurrir al lector estándar.

- `withSchema`— Un valor de cadena que especifica un esquema de tabla en el formato descrito en the section called “Especifique el esquema XML”. Solo se usa con `optimizePerformance` al leer desde conexiones que no son de catálogo.

Uso del lector SIMD JSON vectorizado con formato en columnas de Apache Arrow

AWS Glue versión 3.0 agrega un lector vectorizado para datos JSON. Funciona dos veces más rápido en ciertas condiciones, en comparación con el lector estándar. Este lector viene con ciertas limitaciones documentadas en esta sección que los usuarios deben tener en cuenta antes de usarlo.

Para usar el lector optimizado, configure `"optimizePerformance"` como True en `format_options` o propiedad de tabla. También deberá proporcionar `withSchema` a menos que esté leyendo del catálogo. `withSchema` espera una entrada como se describe en the section called “Especifique el esquema XML”

```
// Read from S3 data source
glueContext.create_dynamic_frame.from_options(
    connection_type = "s3",
    connection_options = {"paths": ["s3://s3path"]},
    format = "json",
    format_options={
        "optimizePerformance": True,
        "withSchema": SchemaString
        })

// Read from catalog table
glueContext.create_dynamic_frame.from_catalog(
    database = database,
    table_name = table,
    additional_options = {
    // The vectorized reader for JSON can read your schema from a catalog table
 property.
        "optimizePerformance": True,
        })
```

Para obtener más información sobre el edificio a *SchemaString*de la biblioteca AWS Glue, consultethe section called “Tipos”.

Limitaciones para el lector CSV vectorizado

Presenta las siguientes limitaciones:

- No se admiten elementos JSON con objetos anidados o valores de matriz. Si se incluye, AWS Glue volverá al lector estándar.
- Se debe proporcionar un esquema, ya sea del Catálogo o con `withSchema`.
- No compatible con `multiLine` ni `jsonPath`. Al proporcionar cualquiera de esas opciones, AWS Glue tendrá que recurrir al lector estándar.
- Proporcionar registros de entrada que no coincidan con el esquema de entrada provocará un error en el lector.
- No se crearán registros de errores.
- No se admiten archivos JSON con caracteres multibytes (como caracteres japoneses o chinos).

Uso del formato ORC en AWS Glue

AWS Glue recupera datos de fuentes y escribe datos en destinos almacenados y transportados en varios formatos de datos. Si los datos se almacenan o se transportan en formato de datos ORC, este documento presenta las características disponibles para utilizar los datos en AWS Glue.

AWS Glue admite el uso del formato ORC. Este formato es un formato de datos basado en columnas y orientado al rendimiento. Para obtener una introducción al formato por parte de la autoridad de normalización, consulte Apache Orc.

Puede utilizar AWS Glue para leer archivos ORC de Amazon S3 y de orígenes de streaming, como también para escribir archivos ORC en Amazon S3. Puede leer y escribir archivos `bzip` y `gzip` que contengan archivos ORC de S3. Debe configurar el comportamiento de compresión en el Parámetros de conexión S3 en lugar de en la configuración que se describe en esta página.

En la siguiente tabla se muestran las operaciones comunes de AWS Glue que admiten la opción de formato ORC.

Leer	Escritura	Lectura de streaming	Grupo de archivos pequeños	Marcadores de trabajo
Compatible	Soportado	Soportado	No se admite	Compatible*

* Compatible con AWS Glue versión 1.0 y posterior

Ejemplo: leer archivos o carpetas ORC de S3

Requisitos previos: necesitará las rutas de S3 (`s3path`) de las carpetas o los archivos ORC que desee leer.

Configuración: en las opciones de la función, especifique `format="orc"`. En sus `connection_options`, utilice la clave `paths` para especificar su `s3path`. Puede configurar la forma en que el lector interactúa con S3 en la `connection_options`. Para obtener más información, consulte Tipos y opciones de conexión para ETL en AWS Glue: the section called "Parámetros de conexión S3".

El siguiente script de ETL de AWS Glue muestra el proceso de lectura de archivos o carpetas ORC de S3:

Python

Para este ejemplo, use el método create_dynamic_frame.from_options.

```
from pyspark.context import SparkContext
from awsglue.context import GlueContext

sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

dynamicFrame = glueContext.create_dynamic_frame.from_options(
    connection_type="s3",
    connection_options={"paths": ["s3://s3path"]},
    format="orc"
)
```

También puede usar DataFrames en un script (`pyspark.sql.DataFrame`).

```
dataFrame = spark.read\
    .orc("s3://s3path")
```

Scala

Para este ejemplo, use la operación getSourceWithFormat.

```
import com.amazonaws.services.glue.util.JsonOptions
import com.amazonaws.services.glue.GlueContext
import org.apache.spark.sql.SparkContext

object GlueApp {
  def main(sysArgs: Array[String]): Unit = {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)

    val dynamicFrame = glueContext.getSourceWithFormat(
      connectionType="s3",
      format="orc",
      options=JsonOptions("""{"paths": ["s3://s3path"]}""")
    ).getDynamicFrame()
  }
}
```

También puede usar DataFrames en un script (`pyspark.sql.DataFrame`).

```
val dataFrame = spark.read
    .orc("s3://s3path")
```

Ejemplo: escribir archivos y carpetas ORC en S3

Requisitos previos: necesitará un DataFrame inicializado (`dataFrame`) o DynamicFrame (`dynamicFrame`). También necesitará la ruta de salida S3 prevista, `s3path`.

Configuración: en las opciones de la función, especifique `format="orc"`. En las opciones de conexión, utilice la clave `paths` para especificar `s3path`. Puede modificar aún más la forma en que el escritor interactúa con S3 en las `connection_options`. Para obtener más información, consulte Opciones de formato de datos para entradas y salidas de ETL en AWS Glue: the section called “Parámetros de conexión S3”. El siguiente ejemplo de código muestra el proceso:

Python

Para este ejemplo, use el método write_dynamic_frame.from_options.

```
from pyspark.context import SparkContext
from awsglue.context import GlueContext
```

```
sc = SparkContext.getOrCreate()
glueContext = GlueContext(sc)

glueContext.write_dynamic_frame.from_options(
    frame=dynamicFrame,
    connection_type="s3",
    format="orc",
    connection_options={
        "path": "s3://s3path"
    }
)
```

También puede usar DataFrames en un script (`pyspark.sql.DataFrame`).

```
df.write.orc("s3://s3path/")
```

Scala

Para este ejemplo, use el método getSinkWithFormat.

```
import com.amazonaws.services.glue.util.JsonOptions
import com.amazonaws.services.glue.{DynamicFrame, GlueContext}
import org.apache.spark.SparkContext

object GlueApp {
  def main(sysArgs: Array[String]): Unit = {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)

    glueContext.getSinkWithFormat(
      connectionType="s3",
      options=JsonOptions("""{"path": "s3://s3path"}"""),
      format="orc"
    ).writeDynamicFrame(dynamicFrame)
  }
}
```

También puede usar DataFrames en un script (`pyspark.sql.DataFrame`).

```
df.write.orc("s3://s3path/")
```

Referencia de configuración de ORC

No hay valores `format_options` para `format="orc"`. No obstante, se le pueden pasar las opciones que acepte el código SparkSQL subyacente mediante el parámetro de mapa `connection_options`.

Uso de marcos de lagos de datos con trabajos de ETL de AWS Glue

Los marcos de lagos de datos de código abierto simplifican el procesamiento incremental de los datos para los archivos que se almacenan en los lagos de datos creados en Amazon S3. AWS Glue 3.0 y versiones posteriores admiten los siguientes marcos de lagos de datos de código abierto:

- Apache Hudi
- Linux Foundation Delta Lake
- Apache Iceberg

Proporcionamos compatibilidad nativa para estos marcos para que pueda leer y escribir los datos que almacena en Amazon S3 de manera coherente desde el punto de vista transaccional. No es necesario instalar ningún conector independiente ni completar pasos de configuración adicionales para utilizar estos marcos en los trabajos de ETL de AWS Glue.

Cuando administre conjuntos de datos mediante AWS Glue Data Catalog, puede utilizar los métodos AWS Glue para leer y escribir tablas de lagos de datos con DataFrames de Spark. También puede leer y escribir datos de Amazon S3 mediante la API de DataFrame de Spark.

En este video, puede obtener información básica sobre el trabajo de Apache Hudi, Apache Iceberg y Delta Lake. Verá cómo insertar, actualizar y eliminar datos en su lago de datos y cómo funciona cada uno de estos marcos.

Temas

- Limitaciones
- Uso del marco de Hudi en AWS Glue
- Uso del marco de Delta Lake en AWS Glue
- Uso del marco de Iceberg en AWS Glue

Limitaciones

Tenga en cuenta las siguientes limitaciones antes de utilizar marcos de data lake. AWS Glue

- Los siguientes AWS Glue `GlueContext` métodos DynamicFrame no admiten la lectura ni la escritura de tablas de marcos de data lake. En su lugar, usa los `GlueContext` DataFrame métodos de nuestra DataFrame API de Spark.
  - Los siguientes `GlueContext` métodos no DynamicFrame son compatibles con el control de permisos de Lake Formation:
    - `create_dynamic_frame.from_catalog`
    - `write_dynamic_frame.from_catalog`
    - `getDynamicFrame`
    - `writeDynamicFrame`
  - El control de permisos de DataFrame Lake Formation admite los siguientes `GlueContext` métodos:
    - `create_data_frame.from_catalog`
    - `write_data_frame.from_catalog`
    - `getDataFrame`
    - `writeDataFrame`
- No se admite la agrupación de archivos pequeños.
- No se admiten los marcadores de trabajo.
- Apache Hudi 0.10.1 para AWS Glue 3.0 no admite Hudi Merge en tablas Read (MoR).
- `ALTER TABLE … RENAME TO`no está disponible para Apache Iceberg 0.13.1 para 3.0. AWS Glue

Limitaciones de las tablas con formato de lago de datos administradas por los permisos de Lake Formation

Los formatos de lago de datos se integran con AWS Glue ETL a través de los permisos de Lake Formation. No `create_dynamic_frame` se admite la creación de un DynamicFrame uso. Para obtener más información, consulte los ejemplos siguientes:

- Ejemplo: Lea y escriba una tabla de iceberg con el control de permisos de Lake Formation
- Ejemplo: Lea y escriba una tabla Hudi con el control de permisos de Lake Formation
- Ejemplo: Lea y escriba la tabla de Delta Lake con el control de permisos de Lake Formation

> **Note**
>
> La integración con AWS Glue ETL mediante los permisos de Lake Formation para Apache Hudi, Apache Iceberg y Delta Lake solo se admite en la AWS Glue versión 4.0.

Apache Iceberg tiene la mejor integración con AWS Glue ETL a través de los permisos de Lake Formation. Es compatible con casi todas las operaciones e incluye soporte para SQL.

Hudi es compatible con la mayoría de las operaciones básicas, con la excepción de las operaciones administrativas. Esto se debe a que estas opciones generalmente se realizan mediante la escritura de marcos de datos y se especifican mediante `additional_options`. Debe utilizar AWS Glue las API DataFrames para crear sus operaciones, ya que SparkSQL no es compatible.

Delta Lake solo admite la lectura, la adición y la sobrescritura de datos de tablas. Delta Lake requiere el uso de sus propias bibliotecas para poder realizar diversas tareas, como las actualizaciones.

Las siguientes funciones no están disponibles para las tablas Iceberg administradas por los permisos de Lake Formation.

- Compactación mediante ETL AWS Glue
- Soporte para Spark SQL a través de AWS Glue ETL

Las siguientes son limitaciones de las tablas Hudi administradas por los permisos de Lake Formation:

- Eliminación de archivos huérfanos

Las siguientes son limitaciones de las tablas de Delta Lake administradas por los permisos de Lake Formation:

- Todas las funciones, excepto la inserción y lectura de las tablas de Delta Lake.

Uso del marco de Hudi en AWS Glue

AWS Glue 3.0 y versiones posteriores son compatibles con el marco de Apache Hudi para lagos de datos. Hudi es un marco de almacenamiento de lagos de datos de código abierto que simplifica el procesamiento progresivo de datos y el desarrollo de canalizaciones de datos. En este tema, se describen las características disponibles para usar los datos en AWS Glue al transportar o almacenar

datos en una tabla de Hudi. Para más información sobre Hudi, consulte la documentación oficial de Apache Hudi.

Puede usar AWS Glue para llevar a cabo operaciones de lectura y escritura en tablas de Hudi en Amazon S3, o trabajar con tablas de Hudi mediante el Catálogo de datos de AWS Glue. También se admiten operaciones adicionales, como insertar, actualizar y todas las operaciones de Apache Spark.

> **Note**
>
> Apache Hudi 0.10.1 para AWS Glue 3.0 no admite las tablas Merge on Read (MoR, fusionar al leer) de Hudi.

La siguiente tabla muestra la versión de Hudi que se incluye en cada versión de AWS Glue.

Versión de AWS Glue	Versión de Hudi compatible
4.0	0.12.1
3.0	0.10.1

Para más información sobre los marcos de lagos de datos compatibles con AWS Glue, consulte Uso de marcos de lagos de datos con trabajos de ETL de AWS Glue.

Activación de Hudi

Para activar Hudi para AWS Glue, haga las siguientes tareas:

- Especifique `hudi` como valor para el parámetro del trabajo `--datalake-formats`. Para obtener más información, consulte Parámetros de los trabajos de AWS Glue.
- Cree una clave con el nombre `--conf` para el trabajo de AWS Glue y establézcala en el siguiente valor. Como alternativa, puede establecer la siguiente configuración mediante `SparkConf` en su script. Esta configuración ayuda a Apache Spark a gestionar correctamente las tablas de Hudi.

```
spark.serializer=org.apache.spark.serializer.KryoSerializer --conf
 spark.sql.hive.convertMetastoreParquet=false
```

- La compatibilidad con los permisos de Lake Formation para Hudi está habilitada de forma predeterminada en AWS Glue 4.0. No se necesita ninguna configuración adicional para leer o

escribir en las tablas Hudi registradas en Lake Formation. Para leer una tabla Hudi registrada, el rol de IAM del trabajo de AWS Glue debe tener el permiso SELECT. Para escribir en una tabla Hudi registrada, el rol de IAM del trabajo de AWS Glue debe tener el permiso SUPER. Para obtener más información sobre la administración de los permisos de Lake Formation, consulte Otorgar y revocar permisos en los recursos del catálogo de datos.

Uso de una versión diferente de Hudi

Para usar una versión de Hudi que no sea compatible con AWS Glue, indique sus propios archivos JAR de Hudi mediante el parámetro de trabajo `--extra-jars`. No incluya `hudi` como valor para el parámetro de trabajo `--datalake-formats`.

Ejemplo: escribir una tabla de Hudi en Amazon S3 y registrarla en el Catálogo de datos de AWS Glue

En este script de ejemplo, se muestra cómo escribir una tabla de Hudi en Amazon S3 y registrar la tabla en el Catálogo de datos de AWS Glue. En el ejemplo, se utiliza la herramienta Hive Sync de Hudi para registrar la tabla.

> **Note**
>
> En este ejemplo, se requiere que establezca el parámetro de trabajo de `--enable-glue-datacatalog` para utilizar el Catálogo de datos de AWS Glue como metaalmacén de Apache Spark Hive. Para obtener más información, consulte Parámetros de los trabajos de AWS Glue.

Python

```
# Example: Create a Hudi table from a DataFrame
# and register the table to Glue Data Catalog

additional_options={
    "hoodie.table.name": "<your_table_name>",
    "hoodie.datasource.write.storage.type": "COPY_ON_WRITE",
    "hoodie.datasource.write.operation": "upsert",
    "hoodie.datasource.write.recordkey.field": "<your_recordkey_field>",
    "hoodie.datasource.write.precombine.field": "<your_precombine_field>",
    "hoodie.datasource.write.partitionpath.field": "<your_partitionkey_field>",
    "hoodie.datasource.write.hive_style_partitioning": "true",
```

```
    "hoodie.datasource.hive_sync.enable": "true",
    "hoodie.datasource.hive_sync.database": "<your_database_name>",
    "hoodie.datasource.hive_sync.table": "<your_table_name>",
    "hoodie.datasource.hive_sync.partition_fields": "<your_partitionkey_field>",
    "hoodie.datasource.hive_sync.partition_extractor_class":
 "org.apache.hudi.hive.MultiPartKeysValueExtractor",
    "hoodie.datasource.hive_sync.use_jdbc": "false",
    "hoodie.datasource.hive_sync.mode": "hms",
    "path": "s3://<s3Path/>"
}

dataFrame.write.format("hudi") \
    .options(**additional_options) \
    .mode("overwrite") \
    .save()
```

Scala

```
// Example: Example: Create a Hudi table from a DataFrame
// and register the table to Glue Data Catalog

val additionalOptions = Map(
  "hoodie.table.name" -> "<your_table_name>",
  "hoodie.datasource.write.storage.type" -> "COPY_ON_WRITE",
  "hoodie.datasource.write.operation" -> "upsert",
  "hoodie.datasource.write.recordkey.field" -> "<your_recordkey_field>",
  "hoodie.datasource.write.precombine.field" -> "<your_precombine_field>",
  "hoodie.datasource.write.partitionpath.field" -> "<your_partitionkey_field>",
  "hoodie.datasource.write.hive_style_partitioning" -> "true",
  "hoodie.datasource.hive_sync.enable" -> "true",
  "hoodie.datasource.hive_sync.database" -> "<your_database_name>",
  "hoodie.datasource.hive_sync.table" -> "<your_table_name>",
  "hoodie.datasource.hive_sync.partition_fields" -> "<your_partitionkey_field>",
  "hoodie.datasource.hive_sync.partition_extractor_class" ->
 "org.apache.hudi.hive.MultiPartKeysValueExtractor",
  "hoodie.datasource.hive_sync.use_jdbc" -> "false",
  "hoodie.datasource.hive_sync.mode" -> "hms",
  "path" -> "s3://<s3Path/>")

dataFrame.write.format("hudi")
  .options(additionalOptions)
  .mode("append")
  .save()
```

Ejemplo: leer una tabla de Hudi de Amazon S3 con el Catálogo de datos de AWS Glue

En este ejemplo, se lee la tabla de Hudi que creó en Ejemplo: escribir una tabla de Hudi en Amazon S3 y registrarla en el Catálogo de datos de AWS Glue de Amazon S3.

> **Note**
>
> En este ejemplo, se requiere que establezca el parámetro de trabajo de `--enable-glue-datacatalog` para utilizar el Catálogo de datos de AWS Glue como metaalmacén de Apache Spark Hive. Para obtener más información, consulte Parámetros de los trabajos de AWS Glue.

Python

Para este ejemplo, utilice el método `GlueContext.create_data_frame.from_catalog()`.

```
# Example: Read a Hudi table from Glue Data Catalog

from awsglue.context import GlueContext
from pyspark.context import SparkContext

sc = SparkContext()
glueContext = GlueContext(sc)

dataFrame = glueContext.create_data_frame.from_catalog(
    database = "<your_database_name>",
    table_name = "<your_table_name>"
)
```

Scala

Para este ejemplo, utilice el método getCatalogSource.

```
// Example: Read a Hudi table from Glue Data Catalog

import com.amazonaws.services.glue.GlueContext
import org.apache.spark.SparkContext

object GlueApp {
  def main(sysArgs: Array[String]): Unit = {
```

```
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)

    val dataFrame = glueContext.getCatalogSource(
      database = "<your_database_name>",
      tableName = "<your_table_name>"
    ).getDataFrame()
  }
}
```

Ejemplo: actualizar e insertar un objeto **`DataFrame`** en una tabla de Hudi en Amazon S3

En este ejemplo, se utiliza el Catálogo de datos de AWS Glue para insertar un DataFrame en la tabla de Hudi que creó en Ejemplo: escribir una tabla de Hudi en Amazon S3 y registrarla en el Catálogo de datos de AWS Glue.

> **Note**
>
> En este ejemplo, se requiere que establezca el parámetro de trabajo de `--enable-glue-datacatalog` para utilizar el Catálogo de datos de AWS Glue como metaalmacén de Apache Spark Hive. Para obtener más información, consulte Parámetros de los trabajos de AWS Glue.

Python

Para este ejemplo, utilice el método `GlueContext.write_data_frame.from_catalog()`.

```
# Example: Upsert a Hudi table from Glue Data Catalog

from awsglue.context import GlueContext
from pyspark.context import SparkContext

sc = SparkContext()
glueContext = GlueContext(sc)

glueContext.write_data_frame.from_catalog(
    frame = dataFrame,
    database = "<your_database_name>",
    table_name = "<your_table_name>",
```

```
    additional_options={
        "hoodie.table.name": "<your_table_name>",
        "hoodie.datasource.write.storage.type": "COPY_ON_WRITE",
        "hoodie.datasource.write.operation": "upsert",
        "hoodie.datasource.write.recordkey.field": "<your_recordkey_field>",
        "hoodie.datasource.write.precombine.field": "<your_precombine_field>",
        "hoodie.datasource.write.partitionpath.field": "<your_partitionkey_field>",
        "hoodie.datasource.write.hive_style_partitioning": "true",
        "hoodie.datasource.hive_sync.enable": "true",
        "hoodie.datasource.hive_sync.database": "<your_database_name>",
        "hoodie.datasource.hive_sync.table": "<your_table_name>",
        "hoodie.datasource.hive_sync.partition_fields": "<your_partitionkey_field>",
        "hoodie.datasource.hive_sync.partition_extractor_class":
 "org.apache.hudi.hive.MultiPartKeysValueExtractor",
        "hoodie.datasource.hive_sync.use_jdbc": "false",
        "hoodie.datasource.hive_sync.mode": "hms"
    }
)
```

Scala

Para este ejemplo, utilice el método getCatalogSink.

```
// Example: Upsert a Hudi table from Glue Data Catalog

import com.amazonaws.services.glue.GlueContext
import com.amazonaws.services.glue.util.JsonOptions
import org.apacke.spark.SparkContext

object GlueApp {
  def main(sysArgs: Array[String]): Unit = {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)
    glueContext.getCatalogSink("<your_database_name>", "<your_table_name>",
      additionalOptions = JsonOptions(Map(
        "hoodie.table.name" -> "<your_table_name>",
        "hoodie.datasource.write.storage.type" -> "COPY_ON_WRITE",
        "hoodie.datasource.write.operation" -> "upsert",
        "hoodie.datasource.write.recordkey.field" -> "<your_recordkey_field>",
        "hoodie.datasource.write.precombine.field" -> "<your_precombine_field>",
        "hoodie.datasource.write.partitionpath.field" ->
 "<your_partitionkey_field>",
        "hoodie.datasource.write.hive_style_partitioning" -> "true",
        "hoodie.datasource.hive_sync.enable" -> "true",
```

```
        "hoodie.datasource.hive_sync.database" -> "<your_database_name>",
        "hoodie.datasource.hive_sync.table" -> "<your_table_name>",
        "hoodie.datasource.hive_sync.partition_fields" ->
 "<your_partitionkey_field>",
        "hoodie.datasource.hive_sync.partition_extractor_class" ->
 "org.apache.hudi.hive.MultiPartKeysValueExtractor",
        "hoodie.datasource.hive_sync.use_jdbc" -> "false",
        "hoodie.datasource.hive_sync.mode" -> "hms"
      )))
      .writeDataFrame(dataFrame, glueContext)
  }
}
```

Ejemplo: leer una tabla de Hudi de Amazon S3 con Spark

En este ejemplo, se lee una tabla de Hudi de Amazon S3 con la API de DataFrame de Spark.

Python

```
# Example: Read a Hudi table from S3 using a Spark DataFrame

dataFrame = spark.read.format("hudi").load("s3://<s3path/>")
```

Scala

```
// Example: Read a Hudi table from S3 using a Spark DataFrame

val dataFrame = spark.read.format("hudi").load("s3://<s3path/>")
```

Ejemplo: escribir una tabla de Hudi en Amazon S3 con Spark

En este ejemplo, se escribe una tabla de Hudi en Amazon S3 con Spark.

Python

```
# Example: Write a Hudi table to S3 using a Spark DataFrame

dataFrame.write.format("hudi") \
    .options(**additional_options) \
    .mode("overwrite") \
```

```
    .save("s3://<s3Path/>)
```

Scala

```
// Example: Write a Hudi table to S3 using a Spark DataFrame

dataFrame.write.format("hudi")
  .options(additionalOptions)
  .mode("overwrite")
  .save("s3://<s3path/>")
```

Ejemplo: Lea y escriba una tabla Hudi con el control de permisos de Lake Formation

En este ejemplo se lee y escribe una tabla Hudi con el control de permisos de Lake Formation.

1. Cree una tabla Hudi y regístrela en Lake Formation.
   a. Para habilitar el control de permisos de Lake Formation, primero tendrá que registrar la tabla de la ruta Amazon S3 en Lake Formation. Para obtener más información, consulte Registro de una ubicación de Amazon S3. Puede registrarlo desde la consola de Lake Formation o mediante la AWS CLI:

      ```
      aws lakeformation register-resource --resource-arn arn:aws:s3:::<s3-bucket>/<s3-
      folder> --use-service-linked-role --region <REGION>
      ```

      Una vez que registre una ubicación de Amazon S3, cualquier tabla de AWS Glue que apunte a la ubicación (o a cualquiera de sus ubicaciones secundarias) devolverá el valor del parámetro `IsRegisteredWithLakeFormation` como verdadero en la llamada `GetTable`.

   b. Cree una tabla Hudi que apunte a la ruta de Amazon S3 registrada a través de la API de marcos de datos de Spark:

      ```
      hudi_options = {
          'hoodie.table.name': table_name,
          'hoodie.datasource.write.storage.type': 'COPY_ON_WRITE',
          'hoodie.datasource.write.recordkey.field': 'product_id',
          'hoodie.datasource.write.table.name': table_name,
          'hoodie.datasource.write.operation': 'upsert',
          'hoodie.datasource.write.precombine.field': 'updated_at',
          'hoodie.datasource.write.hive_style_partitioning': 'true',
          'hoodie.upsert.shuffle.parallelism': 2,
      ```

```
    'hoodie.insert.shuffle.parallelism': 2,
    'path': <S3_TABLE_LOCATION>,
    'hoodie.datasource.hive_sync.enable': 'true',
    'hoodie.datasource.hive_sync.database': database_name,
    'hoodie.datasource.hive_sync.table': table_name,
    'hoodie.datasource.hive_sync.use_jdbc': 'false',
    'hoodie.datasource.hive_sync.mode': 'hms'
}

df_products.write.format("hudi")  \
    .options(**hudi_options)  \
    .mode("overwrite")  \
    .save()
```

2. Conceda permiso a Lake Formation para el rol de IAM del trabajo de AWS Glue. Puede conceder permisos desde la consola de Lake Formation o mediante la AWS CLI. Para obtener más información, consulte Concesión de permisos de tabla mediante la consola de Lake Formation y el método de recurso con nombre
3. Lea la tabla Hudi registrada en Lake Formation. El código es el mismo que leer una tabla Hudi no registrada. Tenga en cuenta que el rol de IAM del trabajo de AWS Glue debe tener el permiso SELECT para que la lectura se realice correctamente.

```
val dataFrame = glueContext.getCatalogSource(
     database = "<your_database_name>",
     tableName = "<your_table_name>"
   ).getDataFrame()
```

4. Escribe en una tabla Hudi registrada en Lake Formation. El código es el mismo que escribir en una tabla Hudi no registrada. Tenga en cuenta que el rol de IAM del trabajo de AWS Glue debe tener el permiso SUPER para que la escritura se realice correctamente.

```
glueContext.getCatalogSink("<your_database_name>", "<your_table_name>",
      additionalOptions = JsonOptions(Map(
        "hoodie.table.name" -> "<your_table_name>",
        "hoodie.datasource.write.storage.type" -> "COPY_ON_WRITE",
        "hoodie.datasource.write.operation" -> "<write_operation>",
        "hoodie.datasource.write.recordkey.field" -> "<your_recordkey_field>",
        "hoodie.datasource.write.precombine.field" -> "<your_precombine_field>",
        "hoodie.datasource.write.partitionpath.field" -> "<your_partitionkey_field>",
        "hoodie.datasource.write.hive_style_partitioning" -> "true",
        "hoodie.datasource.hive_sync.enable" -> "true",
        "hoodie.datasource.hive_sync.database" -> "<your_database_name>",
```

```
        "hoodie.datasource.hive_sync.table" -> "<your_table_name>",
        "hoodie.datasource.hive_sync.partition_fields" ->
"<your_partitionkey_field>",
        "hoodie.datasource.hive_sync.partition_extractor_class" ->
"org.apache.hudi.hive.MultiPartKeysValueExtractor",
        "hoodie.datasource.hive_sync.use_jdbc" -> "false",
        "hoodie.datasource.hive_sync.mode" -> "hms"
      )))
      .writeDataFrame(dataFrame, glueContext)
```

Uso del marco de Delta Lake en AWS Glue

AWS Glue 3.0 y versiones posteriores son compatibles con el marco de Linux Foundation Delta Lake. Delta Lake es un marco de almacenamiento de lagos de datos de código abierto para hacer transacciones de ACID, escalar el manejo de metadatos y unificar la transmisión y el procesamiento de datos por lotes. En este tema se describen las características disponibles para usar los datos en AWS Glue al transportar o almacenar datos en una tabla de Delta Lake. Para más información acerca de Delta Lake, consulte la documentación oficial de Delta Lake.

Puede usar AWS Glue para llevar a cabo operaciones de lectura y escritura en tablas de Delta Lake en Amazon S3, o trabajar con tablas de Delta Lake mediante el Catálogo de datos de AWS Glue. También se admiten operaciones adicionales, como la inserción, la actualización y las lecturas y escrituras de tablas por lotes. Cuando usa tablas de Delta Lake, también tiene la opción de usar métodos de la biblioteca de Python de Delta Lake, como `DeltaTable.forPath`. Para obtener más información sobre la biblioteca de Python de Delta Lake, consulte la documentación de Python de Delta Lake.

En la siguiente tabla, se muestra la versión de Delta Lake que se incluye en cada versión de AWS Glue.

Versión de AWS Glue	Versión de Delta Lake compatible
4.0	2.1.0
3.0	1.0.0

Para más información sobre los marcos de lagos de datos compatibles con AWS Glue, consulte Uso de marcos de lagos de datos con trabajos de ETL de AWS Glue.

Habilitar Delta Lake para AWS Glue

Para activar Delta Lake para AWS Glue, haga las siguientes tareas:

- Especifique `delta` como valor para el parámetro del trabajo `--datalake-formats`. Para obtener más información, consulte Parámetros de los trabajos de AWS Glue.
- Cree una clave con el nombre `--conf` para el trabajo de AWS Glue y establézcala en el siguiente valor. Como alternativa, puede establecer la siguiente configuración mediante `SparkConf` en su script. Esta configuración ayuda a Apache Spark a gestionar correctamente las tablas de Delta Lake.

```
spark.sql.extensions=io.delta.sql.DeltaSparkSessionExtension --conf
 spark.sql.catalog.spark_catalog=org.apache.spark.sql.delta.catalog.DeltaCatalog --
conf
 spark.delta.logStore.class=org.apache.spark.sql.delta.storage.S3SingleDriverLogStore
```

- La compatibilidad con los permisos de Lake Formation para las tablas Delta está habilitada de forma predeterminada en AWS Glue 4.0. No se necesita ninguna configuración adicional para leer o escribir en las tablas Delta registradas en Lake Formation. Para leer una tabla Delta registrada, el rol de IAM del trabajo de AWS Glue debe tener el permiso SELECT. Para escribir en una tabla Delta registrada, el rol de IAM de AWS Glue debe tener el permiso SUPER. Para obtener más información sobre la administración de los permisos de Lake Formation, consulte Otorgar y revocar permisos en los recursos del catálogo de datos.

Uso de una versión diferente de Delta Lake

Para usar una versión de Delta Lake que no sea compatible con AWS Glue, indique sus propios archivos JAR de Delta Lake mediante el parámetro de trabajo `--extra-jars`. No incluya `delta` como valor para el parámetro de trabajo `--datalake-formats`. En este caso, para utilizar la biblioteca de Python de Delta Lake, debe especificar los archivos JAR de la biblioteca mediante el parámetro de trabajo `--extra-py-files`. La biblioteca de Python viene empaquetada en los archivos JAR de Delta Lake.

Ejemplo: escribir una tabla de Delta Lake en Amazon S3 y regístrela en el Catálogo de datos de AWS Glue

El siguiente script ETL de AWS Glue muestra cómo escribir una tabla de Delta Lake en Amazon S3 y registrar la tabla en el Catálogo de datos de AWS Glue.

Python

```
# Example: Create a Delta Lake table from a DataFrame
# and register the table to Glue Data Catalog

additional_options = {
    "path": "s3://<s3Path>"
}
dataFrame.write \
    .format("delta") \
    .options(**additional_options) \
    .mode("append") \
    .partitionBy("<your_partitionkey_field>") \
    .saveAsTable("<your_database_name>.<your_table_name>")
```

Scala

```
// Example: Example: Create a Delta Lake table from a DataFrame
// and register the table to Glue Data Catalog

val additional_options = Map(
  "path" -> "s3://<s3Path>"
)
dataFrame.write.format("delta")
  .options(additional_options)
  .mode("append")
  .partitionBy("<your_partitionkey_field>")
  .saveAsTable("<your_database_name>.<your_table_name>")
```

Ejemplo: leer una tabla de Delta Lake de Amazon S3 con el Catálogo de datos de AWS Glue

El siguiente script de ETL de AWS Glue lee la tabla de Delta Lake que creó en Ejemplo: escribir una tabla de Delta Lake en Amazon S3 y regístrela en el Catálogo de datos de AWS Glue.

Python

Para este ejemplo, utilice el método create_data_frame.from_catalog.

```
# Example: Read a Delta Lake table from Glue Data Catalog

from awsglue.context import GlueContext
```

```
from pyspark.context import SparkContext

sc = SparkContext()
glueContext = GlueContext(sc)

df = glueContext.create_data_frame.from_catalog(
    database="<your_database_name>",
    table_name="<your_table_name>",
    additional_options=additional_options
)
```

Scala

Para este ejemplo, utilice el método getCatalogSource.

```
// Example: Read a Delta Lake table from Glue Data Catalog

import com.amazonaws.services.glue.GlueContext
import org.apacke.spark.SparkContext

object GlueApp {
  def main(sysArgs: Array[String]): Unit = {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)
    val df = glueContext.getCatalogSource("<your_database_name>",
 "<your_table_name>",
      additionalOptions = additionalOptions)
      .getDataFrame()
  }
}
```

Ejemplo: insertar un elemento **`DataFrame`** en una tabla de Delta Lake de Amazon S3 con el Catálogo de datos de AWS Glue

En este ejemplo, se insertan los datos en la tabla de Delta Lake que creó en Ejemplo: escribir una tabla de Delta Lake en Amazon S3 y regístrela en el Catálogo de datos de AWS Glue.

> **Note**
>
> En este ejemplo, se requiere que establezca el parámetro de trabajo de `--enable-glue-datacatalog` para utilizar el Catálogo de datos de AWS Glue como metaalmacén de

Apache Spark Hive. Para obtener más información, consulte Parámetros de los trabajos de AWS Glue.

Python

Para este ejemplo, utilice el método write_data_frame.from_catalog.

```
# Example: Insert into a Delta Lake table in S3 using Glue Data Catalog

from awsglue.context import GlueContext
from pyspark.context import SparkContext

sc = SparkContext()
glueContext = GlueContext(sc)

glueContext.write_data_frame.from_catalog(
    frame=dataFrame,
    database="<your_database_name>",
    table_name="<your_table_name>",
    additional_options=additional_options
)
```

Scala

Para este ejemplo, utilice el método getCatalogSink.

```
// Example: Insert into a Delta Lake table in S3 using Glue Data Catalog

import com.amazonaws.services.glue.GlueContext
import org.apacke.spark.SparkContext

object GlueApp {
  def main(sysArgs: Array[String]): Unit = {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)
    glueContext.getCatalogSink("<your_database_name>", "<your_table_name>",
      additionalOptions = additionalOptions)
      .writeDataFrame(dataFrame, glueContext)
  }
}
```

Ejemplo: leer una tabla de Delta Lake de Amazon S3 con la API de Spark

En este ejemplo, se lee una tabla de Delta Lake de Amazon S3 con la API de Spark.

Python

```
# Example: Read a Delta Lake table from S3 using a Spark DataFrame

dataFrame = spark.read.format("delta").load("s3://<s3path/>")
```

Scala

```
// Example: Read a Delta Lake table from S3 using a Spark DataFrame

val dataFrame = spark.read.format("delta").load("s3://<s3path/>")
```

Ejemplo: escribir una tabla de Delta Lake en Amazon S3 con Spark

En este ejemplo, se escribe una tabla de Delta Lake en Amazon S3 con Spark.

Python

```
# Example: Write a Delta Lake table to S3 using a Spark DataFrame

dataFrame.write.format("delta") \
    .options(**additional_options) \
    .mode("overwrite") \
    .partitionBy("<your_partitionkey_field>") 
    .save("s3://<s3Path>")
```

Scala

```
// Example: Write a Delta Lake table to S3 using a Spark DataFrame

dataFrame.write.format("delta")
  .options(additionalOptions)
  .mode("overwrite")
  .partitionBy("<your_partitionkey_field>")
  .save("s3://<s3path/>")
```

Ejemplo: Lea y escriba la tabla de Delta Lake con el control de permisos de Lake Formation

En este ejemplo se lee y escribe una tabla de Delta Lake con el control de permisos de Lake Formation.

1. Cree una tabla Delta y regístrela en Lake Formation

   a. Para habilitar el control de permisos de Lake Formation, primero tendrá que registrar la tabla de la ruta Amazon S3 en Lake Formation. Para obtener más información, consulte Registro de una ubicación de Amazon S3. Puede registrarlo desde la consola de Lake Formation o mediante la AWS CLI:

   ```
   aws lakeformation register-resource --resource-arn arn:aws:s3:::<s3-bucket>/<s3-folder> --use-service-linked-role --region <REGION>
   ```

   Una vez que registre una ubicación de Amazon S3, cualquier tabla de AWS Glue que apunte a la ubicación (o a cualquiera de sus ubicaciones secundarias) devolverá el valor del parámetro `IsRegisteredWithLakeFormation` como verdadero en la llamada `GetTable`.

   b. Cree una tabla Delta que apunte a la ruta de Amazon S3 registrada a través de Spark:

   > **Note**
   >
   > A continuación, se muestran algunos ejemplos de Python.

   ```
   dataFrame.write \
    .format("delta") \
    .mode("overwrite") \
    .partitionBy("<your_partitionkey_field>") \
    .save("s3://<the_s3_path>")
   ```

   Una vez que los datos se hayan escrito en Amazon S3, utilice el rastreador AWS Glue para crear una nueva tabla de catálogo de Delta. Para obtener más información, consulte Presentación del soporte de tabla nativa de Delta Lake con rastreadores AWS Glue.

   También puede crear la tabla manualmente a través de la API de AWS Glue.`CreateTable`

2. Conceda permiso a Lake Formation para el rol de IAM del trabajo de AWS Glue. Puede conceder permisos desde la consola de Lake Formation o mediante la AWS CLI. Para obtener más

información, consulte Concesión de permisos de tabla mediante la consola de Lake Formation y el método de recurso con nombre

3. Lea la tabla Delta registrada en Lake Formation. El código es el mismo que leer una tabla Delta no registrada. Tenga en cuenta que el rol de IAM del trabajo de AWS Glue debe tener el permiso SELECT para que la lectura se realice correctamente.

```
# Example: Read a Delta Lake table from Glue Data Catalog

df = glueContext.create_data_frame.from_catalog(
    database="<your_database_name>",
    table_name="<your_table_name>",
    additional_options=additional_options
)
```

4. Escribe a una tabla Delta registrada en Lake Formation. El código es el mismo que escribir en una tabla Delta no registrada. Tenga en cuenta que el rol de IAM del trabajo de AWS Glue debe tener el permiso SUPER para que la escritura se realice correctamente.

   De forma predeterminada, AWS Glue usa `Append` como SaveMode. Puede cambiarlo configurando la opción SaveMode en `additional_options`. Para obtener información sobre la compatibilidad con SaveMode en las tablas Delta, consulte Escribir en una tabla.

```
glueContext.write_data_frame.from_catalog(
    frame=dataFrame,
    database="<your_database_name>",
    table_name="<your_table_name>",
    additional_options=additional_options
)
```

Uso del marco de Iceberg en AWS Glue

AWS Glue 3.0 y versiones posteriores son compatibles con el marco de Apache Iceberg para lagos de datos. Iceberg proporciona un formato de tabla de alto rendimiento que funciona igual que una tabla SQL. En este tema, se describen las características disponibles para usar los datos en AWS Glue al transportar o almacenar datos en una tabla de Iceberg. Para más información sobre Iceberg, consulte la documentación oficial de Apache Iceberg.

Puede usar AWS Glue para llevar a cabo operaciones de lectura y escritura en tablas de Iceberg en Amazon S3, o trabajar con tablas de Iceberg mediante el Catálogo de datos de AWS Glue.

También se admiten operaciones adicionales, como insertar, actualizar y todas las consultas de Spark escrituras de Spark.

> **Note**
>
> `ALTER TABLE … RENAME TO` no está disponible para Apache Iceberg 0.13.1 para AWS Glue 3.0.

La siguiente tabla muestra la versión de Iceberg que se incluye en cada versión de AWS Glue.

Versión de AWS Glue	Versión de Iceberg compatible
4.0	1.0.0
3.0	0.13.1

Para más información sobre los marcos de lagos de datos compatibles con AWS Glue, consulte Uso de marcos de lagos de datos con trabajos de ETL de AWS Glue.

Habilitar el marco de Iceberg

Para habilitar Iceberg para AWS Glue, haga las siguientes tareas:

- Especifique `iceberg` como valor para el parámetro del trabajo `--datalake-formats`. Para obtener más información, consulte Parámetros de los trabajos de AWS Glue.
- Cree una clave con el nombre `--conf` para el trabajo de AWS Glue y establézcala en el siguiente valor. Como alternativa, puede establecer la siguiente configuración mediante `SparkConf` en su script. Esta configuración ayuda a Apache Spark a gestionar correctamente las tablas de Iceberg.

```
spark.sql.extensions=org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions
--conf spark.sql.catalog.glue_catalog=org.apache.iceberg.spark.SparkCatalog
--conf spark.sql.catalog.glue_catalog.warehouse=s3://<your-warehouse-dir>/
--conf spark.sql.catalog.glue_catalog.catalog-
impl=org.apache.iceberg.aws.glue.GlueCatalog
--conf spark.sql.catalog.glue_catalog.io-impl=org.apache.iceberg.aws.s3.S3FileIO
```

Si está leyendo o escribiendo en tablas de Iceberg registradas en Lake Formation, añada la siguiente configuración para habilitar la compatibilidad con Lake Formation. Tenga en cuenta que solo AWS Glue 4.0 es compatible con las tablas Iceberg registradas en Lake Formation:

```
--conf spark.sql.catalog.glue_catalog.glue.lakeformation-enabled=true
--conf spark.sql.catalog.glue_catalog.glue.id=<table-catalog-id>
```

Si utiliza AWS Glue 3.0 con Iceberg 0.13.1, debe establecer las siguientes configuraciones adicionales para utilizar el administrador de bloqueos de Amazon DynamoDB y garantizar la transacción atómica. AWS Glue 4.0 usa un bloqueo positivo de forma predeterminada. Para obtener más información, consulte Iceberg AWS Integrations en la documentación oficial de Apache Iceberg.

```
--conf spark.sql.catalog.glue_catalog.lock-
impl=org.apache.iceberg.aws.glue.DynamoLockManager
--conf spark.sql.catalog.glue_catalog.lock.table=<your-dynamodb-table-name>
```

Usar una versión diferente de Iceberg

Para usar una versión de Iceberg que no sea compatible con AWS Glue, indique sus propios archivos JAR de Iceberg mediante el parámetro de trabajo `--extra-jars`. No incluya `iceberg` como valor para el parámetro `--datalake-formats`.

Habilitar el cifrado para tablas de Iceberg

> **Note**
>
> Las tablas de Iceberg tienen sus propios mecanismos para habilitar el cifrado del lado del servidor. Debe habilitar esta configuración además de la configuración de seguridad de Glue AWS.

Para habilitar el cifrado del lado del servidor en las tablas de Iceberg, consulte las instrucciones de la documentación de Iceberg.

Ejemplo: escriba una tabla de Iceberg en Amazon S3 y regístrela en el Catálogo de datos de AWS Glue

En este script de ejemplo, se muestra cómo escribir una tabla de Iceberg en Amazon S3. En el ejemplo se usan integraciones de AWS de Iceberg para registrar la tabla en el Catálogo de datos de AWS Glue.

Python

```
# Example: Create an Iceberg table from a DataFrame
# and register the table to Glue Data Catalog

dataFrame.createOrReplaceTempView("tmp_<your_table_name>")

query = f"""
CREATE TABLE glue_catalog.<your_database_name>.<your_table_name>
USING iceberg
TBLPROPERTIES ("format-version"="2")
AS SELECT * FROM tmp_<your_table_name>
"""
spark.sql(query)
```

Scala

```
// Example: Example: Create an Iceberg table from a DataFrame
// and register the table to Glue Data Catalog

dataFrame.createOrReplaceTempView("tmp_<your_table_name>")

val query = """CREATE TABLE glue_catalog.<your_database_name>.<your_table_name>
USING iceberg
TBLPROPERTIES ("format-version"="2")
AS SELECT * FROM tmp_<your_table_name>
"""
spark.sql(query)
```

Como alternativa, puede escribir una tabla de Iceberg en Amazon S3 y en el Catálogo de datos con los métodos de Spark.

Requisitos previos: deberá disponer de un catálogo para su uso en la biblioteca Iceberg. Cuando se utiliza el AWS catálogo de datos de Glue AWS, Glue lo simplifica. El catálogo de datos de Glue AWS

está preconfigurado para que lo utilicen las bibliotecas de Spark como `glue_catalog`. Las tablas del catálogo de datos se identifican mediante un *`databaseName`* y un *`tableName`*. Para obtener más información sobre el catálogo de datos de Glue AWS, consulte Detección y catalogación de datos.

Si no utiliza el catálogo de datos de Glue AWS, tendrá que aprovisionar un catálogo a través de las API de Spark. Para obtener más información, consulte Configuración de Spark en la documentación de Iceberg.

En este ejemplo, se escribe una tabla de Iceberg en Amazon S3 y el Catálogo de datos con Spark.

Python

```
# Example: Write an Iceberg table to S3 on the Glue Data Catalog

# Create (equivalent to CREATE TABLE AS SELECT)
dataFrame.writeTo("glue_catalog.databaseName.tableName") \
    .tableProperty("format-version", "2") \
    .create()

# Append (equivalent to INSERT INTO)
dataFrame.writeTo("glue_catalog.databaseName.tableName") \
    .tableProperty("format-version", "2") \
    .append()
```

Scala

```
// Example: Write an Iceberg table to S3 on the Glue Data Catalog

// Create (equivalent to CREATE TABLE AS SELECT)
dataFrame.writeTo("glue_catalog.databaseName.tableName")
    .tableProperty("format-version", "2")
    .create()

// Append (equivalent to INSERT INTO)
dataFrame.writeTo("glue_catalog.databaseName.tableName")
    .tableProperty("format-version", "2")
    .append()
```

Ejemplo: leer una tabla de Iceberg de Amazon S3 con el Catálogo de datos de AWS Glue

En este ejemplo, se lee la tabla de Iceberg que creó en Ejemplo: escriba una tabla de Iceberg en Amazon S3 y regístrela en el Catálogo de datos de AWS Glue.

Python

Para este ejemplo, utilice el método `GlueContext.create_data_frame.from_catalog()`.

```
# Example: Read an Iceberg table from Glue Data Catalog

from awsglue.context import GlueContext
from pyspark.context import SparkContext

sc = SparkContext()
glueContext = GlueContext(sc)

df = glueContext.create_data_frame.from_catalog(
    database="<your_database_name>",
    table_name="<your_table_name>",
    additional_options=additional_options
)
```

Scala

Para este ejemplo, utilice el método getCatalogSource.

```
// Example: Read an Iceberg table from Glue Data Catalog

import com.amazonaws.services.glue.GlueContext
import org.apacke.spark.SparkContext

object GlueApp {
  def main(sysArgs: Array[String]): Unit = {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)
    val df = glueContext.getCatalogSource("<your_database_name>",
 "<your_table_name>",
      additionalOptions = additionalOptions)
      .getDataFrame()
  }
}
```

Ejemplo: insertar un **`DataFrame`** en una tabla de Iceberg de Amazon S3 con el Catálogo de datos de AWS Glue

En este ejemplo, se insertan los datos en la tabla de Iceberg que creó en Ejemplo: escriba una tabla de Iceberg en Amazon S3 y regístrela en el Catálogo de datos de AWS Glue.

> **Note**
>
> En este ejemplo, se requiere que establezca el parámetro de trabajo de `--enable-glue-datacatalog` para utilizar el Catálogo de datos de AWS Glue como metaalmacén de Apache Spark Hive. Para obtener más información, consulte Parámetros de los trabajos de AWS Glue.

Python

Para este ejemplo, utilice el método `GlueContext.write_data_frame.from_catalog()`.

```
# Example: Insert into an Iceberg table from Glue Data Catalog

from awsglue.context import GlueContext
from pyspark.context import SparkContext

sc = SparkContext()
glueContext = GlueContext(sc)

glueContext.write_data_frame.from_catalog(
    frame=dataFrame,
    database="<your_database_name>",
    table_name="<your_table_name>",
    additional_options=additional_options
)
```

Scala

Para este ejemplo, utilice el método getCatalogSink.

```
// Example: Insert into an Iceberg table from Glue Data Catalog

import com.amazonaws.services.glue.GlueContext
import org.apacke.spark.SparkContext
```

```
object GlueApp {
  def main(sysArgs: Array[String]): Unit = {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)
    glueContext.getCatalogSink("<your_database_name>", "<your_table_name>",
      additionalOptions = additionalOptions)
      .writeDataFrame(dataFrame, glueContext)
  }
}
```

Ejemplo: leer una tabla de Iceberg de Amazon S3 con Spark

Requisitos previos: deberá disponer de un catálogo para su uso en la biblioteca Iceberg. Cuando se utiliza el AWS catálogo de datos de Glue AWS, Glue lo simplifica. El catálogo de datos de Glue AWS está preconfigurado para que lo utilicen las bibliotecas de Spark como `glue_catalog`. Las tablas del catálogo de datos se identifican mediante un *`databaseName`* y un *`tableName`*. Para obtener más información sobre el catálogo de datos de Glue AWS, consulte Detección y catalogación de datos.

Si no utiliza el catálogo de datos de Glue AWS, tendrá que aprovisionar un catálogo a través de las API de Spark. Para obtener más información, consulte Configuración de Spark en la documentación de Iceberg.

En este ejemplo, se lee una tabla de Iceberg en Amazon S3 del catálogo de datos con Spark.

Python

```
# Example: Read an Iceberg table on S3 as a DataFrame from the Glue Data Catalog

dataFrame = spark.read.format("iceberg").load("glue_catalog.databaseName.tableName")
```

Scala

```
// Example: Read an Iceberg table on S3 as a DataFrame from the Glue Data Catalog

val dataFrame =
 spark.read.format("iceberg").load("glue_catalog.databaseName.tableName")
```

Ejemplo: Lea y escriba una tabla de iceberg con el control de permisos de Lake Formation

En este ejemplo se lee y escribe una tabla de Iceberg con el control de permisos de Lake Formation.

1. Cree una tabla de iceberg y regístrela en Lake Formation:

   a. Para habilitar el control de permisos de Lake Formation, primero tendrá que registrar la tabla de la ruta Amazon S3 en Lake Formation. Para obtener más información, consulte Registro de una ubicación de Amazon S3. Puede registrarlo desde la consola de Lake Formation o mediante la AWS CLI:

   ```
   aws lakeformation register-resource --resource-arn arn:aws:s3:::<s3-bucket>/<s3-folder> --use-service-linked-role --region <REGION>
   ```

   Una vez que registre una ubicación de Amazon S3, cualquier tabla de AWS Glue que apunte a la ubicación (o a cualquiera de sus ubicaciones secundarias) devolverá el valor del parámetro `IsRegisteredWithLakeFormation` como verdadero en la llamada `GetTable`.

   b. Cree una tabla de Iceberg que apunte a la ruta registrada mediante Spark SQL:

   > **Note**
   >
   > A continuación, se muestran algunos ejemplos de Python.

   ```
   dataFrame.createOrReplaceTempView("tmp_<your_table_name>")

   query = f"""
   CREATE TABLE glue_catalog.<your_database_name>.<your_table_name>
   USING iceberg
   AS SELECT * FROM tmp_<your_table_name>
   """
   spark.sql(query)
   ```

   También puede crear la tabla manualmente mediante la API de AWS Glue.`CreateTable` Para obtener más información, consulte Creación de tablas de Apache Iceberg.

2. Conceda permiso a Lake Formation para el rol de IAM. Puede conceder permisos desde la consola de Lake Formation o mediante la AWS CLI. Para obtener más información, consulte: https://docs.aws.amazon.com/lake-formation/latest/dg/granting-table-permissions.html

3. Lea una tabla de Iceberg registrada en Lake Formation. El código es el mismo que leer una tabla de Icebergs no registrada. Tenga en cuenta que el rol de IAM de su trabajo de AWS Glue debe tener el permiso SELECT para que la lectura se realice correctamente.

```
# Example: Read an Iceberg table from the AWS Glue Data Catalog
from awsglue.context import GlueContextfrom pyspark.context import SparkContext

sc = SparkContext()
glueContext = GlueContext(sc)

df = glueContext.create_data_frame.from_catalog(
    database="<your_database_name>",
    table_name="<your_table_name>",
    additional_options=additional_options
)
```

4. Escriba en una tabla de Iceberg registrada en Lake Formation. El código es el mismo que escribir en una tabla Iceberg no registrada. Tenga en cuenta que su rol de IAM del trabajo de AWS Glue debe tener el permiso SUPER para que la escritura se realice correctamente.

```
glueContext.write_data_frame.from_catalog(
    frame=dataFrame,
    database="<your_database_name>",
    table_name="<your_table_name>",
    additional_options=additional_options
)
```

Referencia de configuración compartida

Puede utilizar los siguientes `format_options` valores con cualquier tipo de formato.

- `attachFilename` — Una cadena en el formato adecuado para utilizarla como nombre de columna. Si proporciona esta opción, el nombre del archivo de origen para el registro se adjuntará a él. El valor del parámetro se utilizará como nombre de la columna.
- `attachTimestamp` — Una cadena en el formato adecuado para utilizarla como nombre de columna. Si proporciona esta opción, la hora de modificación del archivo de origen para el registro se adjuntará a él. El valor del parámetro se utilizará como nombre de la columna.

Soporte de AWS Glue Data Catalog para trabajos de Spark SQL

AWS Glue Data Catalog es un catálogo compatible con metaalmacenes de Apache Hive. Puede configurar los trabajos y los puntos de conexión de desarrollo de AWS Glue para que utilicen el Data Catalog como un metaalmacén de Apache Hive externo. De este modo, podrá ejecutar directamente consultas de Apache Spark SQL en las tablas almacenadas en el catálogo de datos. De forma predeterminada, los marcos dinámicos de AWS Glue se integran con Data Catalog. Sin embargo, con esta característica, los trabajos de Spark SQL pueden empezar a utilizar Data Catalog como un metaalmacén de Hive externo.

Esta característica requiere el acceso de red al punto de conexión de la API de AWS Glue Para trabajos de AWS Glue con conexiones ubicadas en subredes privadas, debe configurar un punto de conexión de VPC o una puerta de enlace NAT para proporcionar el acceso a la red. Para obtener más información sobre la configuración del punto de conexión de VPC, consulte Configuración del acceso de red a los almacenes de datos. Para crear una gateway NAT, consulte Gateways NAT en la Guía del usuario de Amazon VPC.

Puede configurar los trabajos y los puntos de enlace de AWS Glue añadiendo el argumento `"--enable-glue-datacatalog": ""` a los argumentos de los trabajos y los puntos de enlace de desarrollo, respectivamente. Al transferir este argumento, se definen ciertas configuraciones en Spark que le permiten obtener acceso a Data Catalog como un metaalmacén de Hive externo. También permite la compatibilidad de Hive en el objeto `SparkSession` creado en el trabajo o punto de enlace de desarrollo de AWS Glue.

Para permitir el acceso a Data Catalog, verifique la casilla Use AWS Glue Data Catalog as the Hive metastore (Utilizar AWS Glue Data Catalog como metastore de Hive) del grupo Catalog options (Opciones de Catalog), que encontrará en la página Add job (Agregar trabajo) o Add endpoint (Agregar punto de conexión) de la consola. Tenga en cuenta que el rol de IAM utilizado para el trabajo o el punto de enlace de desarrollo debe tener permisos `glue:CreateDatabase`. Si no existe una base de datos, se crea una denominada "`default`" en Data Catalog.

Veamos un ejemplo acerca de cómo se puede utilizar esta característica en los trabajos de Spark SQL. En el siguiente ejemplo, se presupone que se han rastreado los conjuntos de datos de legisladores de EE. UU. disponibles en `s3://awsglue-datasets/examples/us-legislators`.

Para serializar o deserializar los datos de las tablas definidas en AWS Glue Data Catalog, Spark SQL necesita la clase Hive SerDe del formato definido en AWS Glue Data Catalog, en la ruta de clase del trabajo de Spark.

AWS Glue distribuye las clases SerDe de algunos formatos comunes. A continuación, se incluyen los enlaces de Amazon S3 de dichas clases:

- JSON
- XML
- Grok

Añada la clase SerDe de JSON como un JAR adicional al punto de enlace de desarrollo. En el caso de los trabajos, puede añadir la clase SerDe utilizando el argumento `--extra-jars` en el campo "arguments". Para obtener más información, consulte Parámetros de los trabajos de AWS Glue.

A continuación, se muestra un ejemplo de un JSON de entrada que crea un punto de enlace de desarrollo con Data Catalog habilitado para Spark SQL.

```
{
    "EndpointName": "Name",
    "RoleArn": "role_ARN",
    "PublicKey": "public_key_contents",
    "NumberOfNodes": 2,
    "Arguments": {
      "--enable-glue-datacatalog": ""
    },
    "ExtraJarsS3Path": "s3://crawler-public/json/serde/json-serde.jar"
}
```

Ahora, puede consultar las tablas creadas a partir del conjunto de datos de los legisladores de EE. UU con Spark SQL.

```
>>> spark.sql("use legislators")
DataFrame[]
>>> spark.sql("show tables").show()
+-----------+-----------------+-----------+
|   database|        tableName|isTemporary|
+-----------+-----------------+-----------+
|legislators|       areas_json|      false|
|legislators|   countries_json|      false|
|legislators|      events_json|      false|
|legislators| memberships_json|      false|
|legislators|organizations_json|      false|
```

```
|legislators|      persons_json|      false|
+-----------+------------------+-----------+
>>> spark.sql("describe memberships_json").show()
+--------------------+---------+-----------------+
|            col_name|data_type|          comment|
+--------------------+---------+-----------------+
|             area_id|   string|from deserializer|
|     on_behalf_of_id|   string|from deserializer|
|     organization_id|   string|from deserializer|
|                role|   string|from deserializer|
|           person_id|   string|from deserializer|
|legislative_perio...|   string|from deserializer|
|          start_date|   string|from deserializer|
|            end_date|   string|from deserializer|
+--------------------+---------+-----------------+
```

Si la clase SerDe del formato no está disponible en el classpath del trabajo, aparecerá un error similar al siguiente.

```
>>> spark.sql("describe memberships_json").show()

Caused by: MetaException(message:java.lang.ClassNotFoundException Class
 org.openx.data.jsonserde.JsonSerDe not found)
    at
 org.apache.hadoop.hive.metastore.MetaStoreUtils.getDeserializer(MetaStoreUtils.java:399)
    at
 org.apache.hadoop.hive.ql.metadata.Table.getDeserializerFromMetaStore(Table.java:276)
    ... 64 more
```

Para ver únicamente los diferentes valores de `organization_id` de la tabla `memberships`, ejecute la siguiente consulta SQL.

```
>>> spark.sql("select distinct organization_id from memberships_json").show()
+--------------------+
|     organization_id|
+--------------------+
|d56acebe-8fdc-47b...|
|8fa6c3d2-71dc-478...|
+--------------------+
```

Si debe hacer lo mismo con marcos dinámicos, ejecute lo siguiente.

```
>>> memberships = glueContext.create_dynamic_frame.from_catalog(database="legislators",
 table_name="memberships_json")
>>> memberships.toDF().createOrReplaceTempView("memberships")
>>> spark.sql("select distinct organization_id from memberships").show()
+--------------------+
|     organization_id|
+--------------------+
|d56acebe-8fdc-47b...|
|8fa6c3d2-71dc-478...|
+--------------------+
```

Aunque DynamicFrames está optimizado para las operaciones de ETL, permitir que Spark SQL tenga acceso a Data Catalog es un modo sencillo de ejecutar instrucciones SQL complejas o migrar aplicaciones existentes.

Uso de marcadores de trabajo

AWS Glue para Spark usa marcadores de flujo de trabajo para realizar un seguimiento de los datos que ya se han procesado. Para obtener un resumen de la característica de marcadores de trabajo y lo que admite, consulte the section called “Seguimiento de los datos procesados mediante marcadores de trabajo”. Al programar un trabajo de AWS Glue con marcadores, tiene acceso a una flexibilidad que no está disponible en los trabajos visuales.

- Al leer desde la JDBC, puede especificar las columnas que se van a utilizar como claves de marcador en el script de AWS Glue.
- Puede elegir qué `transformation_ctx` aplicar a cada llamada de método.

Siempre, al principio de un script, haga una llamada a *`job.init`* y, al final, a *`job.commit`* con los parámetros configurados de manera apropiada. Estas dos características se utilizan para inicializar el servicio de marcadores y actualizar el servicio con el cambio de estado. Los favoritos no funcionarán sin llamarlos.

Especifique las claves de los marcadores

En el caso de los flujos de trabajo de JDBC, el marcador realiza un seguimiento de las filas que ha leído el trabajo al comparar los valores de los campos clave con un valor marcado. Esto no es necesario ni aplicable a los flujos de trabajo de Amazon S3. Al escribir un script AWS Glue sin el editor visual, puede especificar con marcadores qué columna desea rastrear. También puede

especificar varias columnas. Se permiten espacios en la secuencia de valores al especificar las claves de marcadores definidas por el usuario.

> **Warning**
>
> Si se utilizan claves de marcador definidas por el usuario, deben aumentar o disminuir de forma monotónica estrictamente. Al seleccionar campos adicionales para una clave compuesta, los campos para conceptos como “versiones secundarias” o “números de revisión” no cumplen este criterio, ya que sus valores se reutilizan en todo el conjunto de datos.

Puede especificar `jobBookmarkKeys` y `jobBookmarkKeysSortOrder` de las siguientes maneras:

- `create_dynamic_frame.from_catalog`: utilice `additional_options`.
- `create_dynamic_frame.from_options`: utilice `connection_options`.

Contexto de transformación

Muchos de los métodos de marco dinámico de AWS Glue PySpark incluyen un parámetro opcional denominado `transformation_ctx`, que es un identificador único para la instancia de operador de ETL. El parámetro `transformation_ctx` se utiliza para identificar la información de estado dentro de un marcador de trabajo para el operador determinado. En concreto, AWS Glue utiliza `transformation_ctx` para indexar la clave del estado de marcador.

> **Warning**
>
> `transformation_ctx` sirve de clave para realizar búsquedas en el estado de favorito de un origen específico del script. Para que el favorito funcione correctamente, siempre se debe conservar la coherencia entre el origen y el elemento `transformation_ctx` asociado. Modificar la propiedad de origen o cambiarle el nombre a `transformation_ctx` puede hacer que el favorito anterior deje de ser válido y que el filtrado basado en marcas temporales no produzca el resultado correcto.

Para que los marcadores de flujo de trabajo funcionen correctamente, habilite el parámetro de marcador de trabajo y establezca el parámetro `transformation_ctx`. Si no pasa el parámetro `transformation_ctx`, no se habilitan los marcadores de trabajo para un marco dinámico o una

tabla que se usen en el método. Por ejemplo, si tiene un trabajo de ETL que lee y combina dos orígenes de Amazon S3, puede decidir transferir el parámetro `transformation_ctx` únicamente a los métodos para los que desea habilitar marcadores. Si restablece el marcador de trabajo de un trabajo, se restablecen todas las transformaciones que están asociadas al trabajo sea cual sea el `transformation_ctx` utilizado.

Para obtener más información acerca de la clase `DynamicFrameReader`, consulte DynamicFrameReader clase. Para obtener más información acerca de las extensiones de PySpark, consulte Referencia de las extensiones de PySpark de AWS Glue.

Ejemplos

Example

A continuación, se muestra un ejemplo de un script generado para un origen de datos de Amazon S3. Las partes del script que son necesarias para utilizar marcadores de trabajo se muestran en cursiva. Para obtener más información sobre estos elementos, consulte GlueContext clase API y Clase DynamicFrameWriter API.

```
# Sample Script
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job

args = getResolvedOptions(sys.argv, ['JOB_NAME'])
sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args['JOB_NAME'], args)

datasource0 = glueContext.create_dynamic_frame.from_catalog(
    database = "database",
    table_name = "relatedqueries_csv",
    transformation_ctx = "datasource0"
)

applymapping1 = ApplyMapping.apply(
    frame = datasource0,
```

```
    mappings = [("col0", "string", "name", "string"), ("col1", "string", "number",
 "string")],
    transformation_ctx = "applymapping1"
)

datasink2 = glueContext.write_dynamic_frame.from_options(
    frame = applymapping1,
    connection_type = "s3",
    connection_options = {"path": "s3://input_path"},
    format = "json",
    transformation_ctx = "datasink2"
)


job.commit()
```

Example

A continuación se muestra un ejemplo de un script generado para un origen JDBC. La tabla de origen es una tabla de empleados con la columna empno como clave principal. Aunque el trabajo utiliza una clave principal secuencial como clave de marcador predeterminada, si no se especifica ninguna clave de marcador, porque empno no es necesariamente secuencial (podría haber brechas en los valores), esta clave no califica como clave de marcador predeterminada. Por lo tanto, el script designa empno explícitamente como la clave de marcador. Esa parte del código se muestra en cursiva.

```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job

args = getResolvedOptions(sys.argv, ['JOB_NAME'])

sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args['JOB_NAME'], args)

datasource0 = glueContext.create_dynamic_frame.from_catalog(
```

```
    database = "hr",
    table_name = "emp",
    transformation_ctx = "datasource0",
    additional_options = {"jobBookmarkKeys":["empno"],"jobBookmarkKeysSortOrder":"asc"}
)

applymapping1 = ApplyMapping.apply(
    frame = datasource0,
    mappings = [("ename", "string", "ename", "string"), ("hrly_rate", "decimal(38,0)",
 "hrly_rate", "decimal(38,0)"), ("comm", "decimal(7,2)", "comm", "decimal(7,2)"),
 ("hiredate", "timestamp", "hiredate", "timestamp"), ("empno", "decimal(5,0)", "empno",
 "decimal(5,0)"), ("mgr", "decimal(5,0)", "mgr", "decimal(5,0)"), ("photo", "string",
 "photo", "string"), ("job", "string", "job", "string"), ("deptno", "decimal(3,0)",
 "deptno", "decimal(3,0)"), ("ssn", "decimal(9,0)", "ssn", "decimal(9,0)"), ("sal",
 "decimal(7,2)", "sal", "decimal(7,2)")],
    transformation_ctx = "applymapping1"
)

datasink2 = glueContext.write_dynamic_frame.from_options(
    frame = applymapping1,
    connection_type = "s3",
    connection_options = {"path": "s3://hr/employees"},
    format = "csv",
    transformation_ctx = "datasink2"
)

job.commit()
```

Uso de la detección de datos confidenciales fuera de AWS Glue Studio

AWS Glue Studio te permite detectar datos confidenciales; sin embargo, también puedes usar la funcionalidad de detección de datos confidenciales fuera de AWS Glue Studio.

Para obtener una lista completa de los tipos de datos confidenciales administrados, consulte Tipos de datos administrados.

Detección de datos confidenciales mediante tipos de PII AWS gestionados

AWS Glue proporciona dos API en un trabajo de AWS Glue ETL. Son `detect()` y `classifyColumns()`:

```
  detect(frame: DynamicFrame,
```

```
      entityTypesToDetect: Seq[String],
      outputColumnName: String = "DetectedEntities",
      detectionSensitivity: String = "LOW"): DynamicFrame

  detect(frame: DynamicFrame,
      detectionParameters: JsonOptions,
      outputColumnName: String = "DetectedEntities",
      detectionSensitivity: String = "LOW"): DynamicFrame

   classifyColumns(frame: DynamicFrame,
      entityTypesToDetect: Seq[String],
      sampleFraction: Double = 0.1,
      thresholdFraction: Double = 0.1,
      detectionSensitivity: String = "LOW")
```

Puede usar la `detect()` API para identificar los tipos de PII AWS gestionados y los tipos de entidades personalizadas. Se crea automáticamente una nueva columna con el resultado de la detección. La API `classifyColumns()` devuelve un mapa donde las claves son los nombres de las columnas y los valores son una lista de los tipos de entidades detectadas. `SampleFraction` indica la fracción de los datos que deben cumplirse para que una columna se identifique en busca de entidades de PII, mientras que `ThresholdFraction` indica la fracción de los datos que deben cumplirse para que una columna se identifique como datos de PII.

Detección a nivel de fila

En el ejemplo, el trabajo consiste en realizar las siguientes acciones mediante las API `detect()` y `classifyColumns()`:

- leer datos de un Amazon S3 depósito y convertirlos en un DynamicFrame
- detectar instancias “Email” (Correo electrónico) y “Credit Card” (Tarjeta de crédito) en el dynamicFrame
- devolver un dynamicFrame con los valores originales más una columna que incluye el resultado de la detección para cada fila
- escribir el DynamicFrame devuelto en otra ruta Amazon S3

```
import com.amazonaws.services.glue.GlueContext
import com.amazonaws.services.glue.MappingSpec
```

```
 import com.amazonaws.services.glue.errors.CallSite
 import com.amazonaws.services.glue.util.GlueArgParser
 import com.amazonaws.services.glue.util.Job
 import com.amazonaws.services.glue.util.JsonOptions
 import org.apache.spark.SparkContext
 import scala.collection.JavaConverters._
 import com.amazonaws.services.glue.ml.EntityDetector

 object GlueApp {
   def main(sysArgs: Array[String]) {
     val spark: SparkContext = new SparkContext()
     val glueContext: GlueContext = new GlueContext(spark)
     val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB_NAME").toArray)
     Job.init(args("JOB_NAME"), glueContext, args.asJava)
     val frame=
glueContext.getSourceWithFormat(formatOptions=JsonOptions("""{"quoteChar": "\"",
"withHeader": true, "separator": ","}"""), connectionType="s3", format="csv",
options=JsonOptions("""{"paths": ["s3://pathToSource"], "recurse": true}"""),
transformationContext="AmazonS3_node1650160158526").getDynamicFrame()

     val frameWithDetectedPII = EntityDetector.detect(frame, Seq("EMAIL",
"CREDIT_CARD"))

     glueContext.getSinkWithFormat(connectionType="s3",
options=JsonOptions("""{"path": "s3://pathToOutput/", "partitionKeys": []}"""),
transformationContext="someCtx",
format="json").writeDynamicFrame(frameWithDetectedPII)

     Job.commit()
   }
 }
```

Detección a nivel de fila con acciones detalladas

En el ejemplo, el trabajo consiste en realizar las siguientes acciones mediante las API de `detect()`:

- leer datos de un bucket de Amazon S3 y convertirlos en un dynamicFrame
- detectar tipos de datos confidenciales para “USA_PTIN”, “BANK_ACCOUNT”, “USA_SSN”, “USA_PASSPORT_NUMBER” y “PHONE_NUMBER” en el dynamicFrame
- devolver un dynamicFrame con los valores modificados enmascarados más una columna que incluye el resultado de la detección para cada fila

- escribir el dynamicFrame devuelto en otra ruta de Amazon S3

A diferencia de la API `detect()` anterior, esta utiliza acciones detalladas para detectar los tipos de entidades. Para obtener más información, consulte Parámetros de detección para su uso de `detect()`.

```
import com.amazonaws.services.glue.GlueContext
import com.amazonaws.services.glue.MappingSpec
import com.amazonaws.services.glue.errors.CallSite
import com.amazonaws.services.glue.util.GlueArgParser
import com.amazonaws.services.glue.util.Job
import com.amazonaws.services.glue.util.JsonOptions
import org.apache.spark.SparkContext
import scala.collection.JavaConverters._
import com.amazonaws.services.glue.ml.EntityDetector

object GlueApp {
  def main(sysArgs: Array[String]) {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)
    val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB_NAME").toArray)
    Job.init(args("JOB_NAME"), glueContext, args.asJava)
    val frame =
 glueContext.getSourceWithFormat(formatOptions=JsonOptions("""{"quoteChar": "\"",
 "withHeader": true, "separator": ","}"""), connectionType="s3", format="csv",
 options=JsonOptions("""{"paths": ["s3://pathToSource"], "recurse": true}"""),
 transformationContext="AmazonS3_node_source").getDynamicFrame()

    val detectionParameters = JsonOptions(
      """
        {
          "USA_DRIVING_LICENSE": [{
            "action": "PARTIAL_REDACT",
            "sourceColumns": ["Driving License"],
            "actionOptions": {
              "matchPattern": "[0-9]",
              "redactChar": "*"
            }
          }],
          "BANK_ACCOUNT": [{
            "action": "DETECT",
            "sourceColumns": ["*"]
```

```
          }],
          "USA_SSN": [{
            "action": "SHA256_HASH",
            "sourceColumns": ["SSN"]
          }],
          "IP_ADDRESS": [{
            "action": "REDACT",
            "sourceColumns": ["IP Address"],
            "actionOptions": {"redactText": "*****"}
          }],
          "PHONE_NUMBER": [{
            "action": "PARTIAL_REDACT",
            "sourceColumns": ["Phone Number"],
            "actionOptions": {
              "numLeftCharsToExclude": 1,
              "numRightCharsToExclude": 0,
              "redactChar": "*"
            }
          }]
        }
      """
    )

    val frameWithDetectedPII = EntityDetector.detect(frame, detectionParameters,
 "DetectedEntities", "HIGH")

    glueContext.getSinkWithFormat(connectionType="s3", options=JsonOptions("""{"path":
 "s3://pathToOutput/", "partitionKeys": []}"""),
 transformationContext="AmazonS3_node_target",
 format="json").writeDynamicFrame(frameWithDetectedPII)

    Job.commit()
  }
}
```

Detección a nivel de columna

En el ejemplo, el trabajo consiste en realizar las siguientes acciones mediante las API de `classifyColumns()`:

- leer datos de un bucket de Amazon S3 y convertirlos en un dynamicFrame

- detectar instancias “Email” (Correo electrónico) y “Credit Card” (Tarjeta de crédito) en el dynamicFrame
- definir los parámetros para realizar una muestra del 100 % de la columna, marcar una entidad como detectada si se encuentra en el 10 % de las células y tener una sensibilidad “BAJA”
- devolver un mapa donde las claves son los nombres de las columnas y los valores son una lista de los tipos de entidades detectadas
- escribir el dynamicFrame devuelto en otra ruta de Amazon S3

```
import com.amazonaws.services.glue.GlueContext
import com.amazonaws.services.glue.MappingSpec
import com.amazonaws.services.glue.errors.CallSite
import com.amazonaws.services.glue.util.GlueArgParser
import com.amazonaws.services.glue.util.Job
import com.amazonaws.services.glue.util.JsonOptions
import org.apache.spark.SparkContext
import scala.collection.JavaConverters._
import com.amazonaws.services.glue.DynamicFrame
import com.amazonaws.services.glue.ml.EntityDetector

object GlueApp {
  def main(sysArgs: Array[String]) {
    val spark: SparkContext = new SparkContext()
    val glueContext: GlueContext = new GlueContext(spark)
    val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB_NAME").toArray)
    Job.init(args("JOB_NAME"), glueContext, args.asJava)
    val frame =
 glueContext.getSourceWithFormat(formatOptions=JsonOptions("""{"quoteChar":
 "\"", "withHeader": true, "separator": ",", "optimizePerformance": false}"""),
 connectionType="s3", format="csv", options=JsonOptions("""{"paths": ["s3://
pathToSource"], "recurse": true}"""), transformationContext="frame").getDynamicFrame()

    import glueContext.sparkSession.implicits._

    val detectedDataFrame = EntityDetector.classifyColumns(
        frame,
        entityTypesToDetect = Seq("CREDIT_CARD", "PHONE_NUMBER"),
        sampleFraction = 1.0,
        thresholdFraction = 0.1,
        detectionSensitivity = "LOW"
    )
    val detectedDF = (detectedDataFrame).toSeq.toDF("columnName", "entityTypes")
```

```
    val DetectSensitiveData_node = DynamicFrame(detectedDF, glueContext)

    glueContext.getSinkWithFormat(connectionType="s3", options=JsonOptions("""{"path":
 "s3://pathToOutput", "partitionKeys": []}"""), transformationContext="someCtx",
 format="json").writeDynamicFrame(DetectSensitiveData_node)

    Job.commit()
  }
}
```

Detección de datos confidenciales: detección mediante AWS CustomEntityType tipos de PII

Puede definir entidades personalizadas a través de AWS Studio. Sin embargo, para utilizar esta función fuera de AWS Studio, primero debe definir los tipos de entidades personalizadas y, a continuación, añadir los tipos de entidades personalizadas definidas a la lista de`entityTypesToDetect`.

Si tiene tipos de datos confidenciales específicos en sus datos (como “Employee ID” [ID de empleado]), puede crear entidades personalizadas si llama a la API `CreateCustomEntityType()`. El siguiente ejemplo define el tipo de entidad personalizado “EMPLOYEE_ID” (Id de empleado) para la API `CreateCustomEntityType()` con los parámetros de solicitud:

```
{
    "name": "EMPLOYEE_ID",
    "regexString": "\d{4}-\d{3}",
    "contextWords": ["employee"]
}
```

A continuación, modifique el trabajo para usar el nuevo tipo de datos confidenciales personalizado al agregar el tipo de entidad personalizado (EMPLOYEE_ID) a la API `EntityDetector()`:

```
import com.amazonaws.services.glue.GlueContext
import com.amazonaws.services.glue.MappingSpec
import com.amazonaws.services.glue.errors.CallSite
import com.amazonaws.services.glue.util.GlueArgParser
import com.amazonaws.services.glue.util.Job
```

```
 import com.amazonaws.services.glue.util.JsonOptions
 import org.apache.spark.SparkContext
 import scala.collection.JavaConverters._
 import com.amazonaws.services.glue.ml.EntityDetector

 object GlueApp {
   def main(sysArgs: Array[String]) {
     val spark: SparkContext = new SparkContext()
     val glueContext: GlueContext = new GlueContext(spark)
     val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB_NAME").toArray)
     Job.init(args("JOB_NAME"), glueContext, args.asJava)
     val frame=
glueContext.getSourceWithFormat(formatOptions=JsonOptions("""{"quoteChar": "\"",
"withHeader": true, "separator": ","}"""), connectionType="s3", format="csv",
options=JsonOptions("""{"paths": ["s3://pathToSource"], "recurse": true}"""),
transformationContext="AmazonS3_node1650160158526").getDynamicFrame()

     val frameWithDetectedPII = EntityDetector.detect(frame, Seq("EMAIL",
"CREDIT_CARD", "EMPLOYEE_ID"))

     glueContext.getSinkWithFormat(connectionType="s3",
options=JsonOptions("""{"path": "s3://pathToOutput/", "partitionKeys": []}"""),
transformationContext="someCtx",
format="json").writeDynamicFrame(frameWithDetectedPII)

     Job.commit()
   }
 }
```

> **Note**
>
> Si se define un tipo de datos confidenciales personalizado con el mismo nombre que un tipo de entidad gestionada existente, el tipo de datos confidenciales personalizado tendrá prioridad y sobrescribirá la lógica del tipo de entidad gestionada.

Parámetros de detección para su uso de **`detect()`**

Este método se utiliza para detectar entidades en un DynamicFrame. Devuelve una columna nueva DataFrame con los valores originales y una columna adicional outputColumnName que contiene metadatos de detección de PII. El enmascaramiento personalizado se puede realizar después de

que DynamicFrame se devuelva en el AWS Glue script o, en su lugar, se puede utilizar la API detect () con acciones detalladas.

```
detect(frame: DynamicFrame,
       entityTypesToDetect: Seq[String],
       outputColumnName: String = "DetectedEntities",
       detectionSensitivity: String = "LOW"): DynamicFrame
```

Parámetros:

- frame: (tipo:`DynamicFrame`) La entrada DynamicFrame que contiene los datos que se van a procesar.
- entityTypesToDetectar: (tipo:`[Seq[String]`) Lista de tipos de entidades que se van a detectar. Pueden ser tipos de entidad administrados o tipos de entidad personalizados.
- outputColumnName— (tipo:`String`, predeterminado: "DetectedEntities«) El nombre de la columna en la que se almacenarán las entidades detectadas. Si no se proporciona, el nombre de columna predeterminado es "DetectedEntities».
- detectionSensitivity – (tipo: `String`, opciones: “LOW” o “HIGH”, predeterminado: “LOW”): especifica la sensibilidad del proceso de detección. Las opciones válidas son “BAJA” o “ALTA”. Si no se proporciona, la sensibilidad predeterminada se establece en “LOW”.

Configuración de `outputColumnName`:

El nombre de la columna en la que se van a almacenar las entidades detectadas. Si no se proporciona, el nombre de columna predeterminado es "DetectedEntities». Para cada fila de la columna de salida, la columna complementaria incluye un mapa del nombre de la columna con los metadatos de la entidad detectada con los siguientes pares clave-valor:

- entityType: el tipo de entidad detectada.
- start: la posición inicial de la entidad detectada en los datos originales.
- final: la posición final de la entidad detectada en los datos originales.
- actionUsed: la acción realizada en la entidad detectada (por ejemplo, "DETECT", "REDACT", "PARTIAL_REDACT", "SHA256_HASH").

Ejemplo:

```
{
   "DetectedEntities":{
      "SSN Col":[
         {
            "entityType":"USA_SSN",
            "actionUsed":"DETECT",
            "start":4,
            "end":15
         }
      ],
      "Random Data col":[
         {
            "entityType":"BANK_ACCOUNT",
            "actionUsed":"PARTIAL_REDACT",
            "start":4,
            "end":13
         },
         {
            "entityType":"IP_ADDRESS",
            "actionUsed":"REDACT",
            "start":4,
            "end":13
         }
      ]
   }
}
```

Parámetros de detección para **`detect()`** con acciones detalladas

Este método se utiliza para detectar entidades en un DynamicFrame mediante parámetros específicos. Devuelve una columna nueva DataFrame con los valores originales reemplazados por datos confidenciales enmascarados y una columna adicional `outputColumnName` que contiene metadatos de detección de información personal.

```
detect(frame: DynamicFrame,
       detectionParameters: JsonOptions,
       outputColumnName: String = "DetectedEntities",
       detectionSensitivity: String = "LOW"): DynamicFrame
```

Parámetros:

- frame — (tipo:`DynamicFrame`): la entrada DynamicFrame que contiene los datos que se van a procesar.
- detectionParameters – (tipo: `Json0ptions`): opciones de JSON que especifican los parámetros del proceso de detección.
- outputColumnName— (tipo:`String`, predeterminado: "DetectedEntities«): nombre de la columna en la que se almacenarán las entidades detectadas. Si no se proporciona, el nombre de columna predeterminado es "DetectedEntities».
- detectionSensitivity – (tipo: `String`, opciones: "LOW" o "HIGH", predeterminado: “LOW”): especifica la sensibilidad del proceso de detección. Las opciones válidas son “BAJA” o “ALTA”. Si no se proporciona, la sensibilidad predeterminada se establece en “BAJA”.

Configuración de `detectionParameters`

Si no se incluye ninguna configuración, se utilizarán los valores predeterminados.

- action – (tipo: `String`, opciones: "DETECT", "REDACT", "PARTIAL_REDACT", "SHA256_HASH"): especifica la acción que se va a realizar en la entidad. Obligatorio. Tenga en cuenta que las acciones que enmascaran (todas excepto “DETECT”) solo pueden realizar una acción por columna. Se trata de una medida preventiva para enmascarar las entidades fusionadas.
- sourceColumns – (tipo: `List[String]`, predeterminado: [“*”]): lista de nombres de columnas de origen para realizar la detección de la entidad. El valor predeterminados es [“*”] si no está presente Eleva `IllegalArgumentException` si se utiliza un nombre de columna no válido.
- sourceColumnsToExcluir: (escriba:`List[String]`) Lista de nombres de columnas de origen para realizar la detección de la entidad. Utilice una de las dos opciones: `sourceColumns` o `sourceColumnsToExclude`. Eleva `IllegalArgumentException` si se utiliza un nombre de columna no válido.
- actionOptions: opciones adicionales basadas en la acción especificada:
  - Para "DETECT" y "SHA256_HASH", no se permiten opciones.
  - Para "REDACT":
    - redactText – (tipo: `String`, predeterminado: “*****”): texto que reemplaza a la entidad detectada.
  - Para “PARTIAL_REDACT”:
    - redactChar – (tipo: `String`, predeterminado: “*”): carácter que reemplaza a todos los caracteres detectados en la entidad.

- matchPattern – (tipo: `String`): patrón de expresiones regulares para redacción parcial. No se puede combinar con numLeftChars ToExclude o`numRightCharsToExclude`.
- numLeftCharsToExclude— (tipo:`String, integer`) Número de caracteres de la izquierda que se van a excluir. No se puede combinar con matchPattern, pero se puede usar con `numRightCharsToExclude`.
- numRightCharsToExclude— (tipo:`String, integer`) Número de caracteres de la derecha que se van a excluir. No se puede combinar con matchPattern, pero se puede usar con `numRightCharsToExclude`.

Configuración de `outputColumnName`

Consulte outputColumnName la configuración

Parámetros de detección para **`classifyColumns()`**

Este método se utiliza para detectar entidades en un DynamicFrame. Devuelve un mapa donde las claves son los nombres de las columnas y los valores son una lista de los tipos de entidades detectadas. El enmascaramiento personalizado se puede realizar una vez que se devuelva en el script de AWS Glue.

```
classifyColumns(frame: DynamicFrame,
                entityTypesToDetect: Seq[String],
                sampleFraction: Double = 0.1,
                thresholdFraction: Double = 0.1,
                detectionSensitivity: String = "LOW")
```

Parámetros:

- frame: (tipo:`DynamicFrame`) La entrada DynamicFrame que contiene los datos que se van a procesar.
- entityTypesToDetectar: (tipo:`Seq[String]`) Lista de tipos de entidades que se van a detectar. Pueden ser tipos de entidad administrados o tipos de entidad personalizados.
- sampleFraction – (tipo: `Double`, predeterminado: 10 %): la fracción de los datos de los que se realizará una muestra al buscar entidades de PII.
- thresholdFraction – (tipo:`Double`, predeterminado: 10 %): la fracción de los datos que deben cumplirse para que una columna se identifique como datos de PII.

- detectionSensitivity – (tipo: `String`, opciones: “LOW” o “HIGH”, predeterminado: “LOW”): especifica la sensibilidad del proceso de detección. Las opciones válidas son “BAJA” o “ALTA”. Si no se proporciona, la sensibilidad predeterminada se establece en “BAJA”.

Tipos de datos confidenciales administrados

Entidades globales

Tipo de datos	Categoría	Descripción
PERSON_NAME	Universal	El nombre de la persona.
CORREO ELECTRÓNICO	Personal	La dirección de correo electrónico.
IP_ADDRESS	Computadora	La dirección IP
MAC_ADDRESS	Personal	La dirección MAC.

Tipos de datos para EE. UU.

Tipo de datos	Descripción
BANK_ACCOUNT	El número de cuenta bancaria. No es específico de un país o región, sin embargo, solo se detectan los formatos de cuenta de EE. UU. y Canadá.
CREDIT_CARD	El número de tarjeta de crédito.
PHONE_NUMBER	El número de teléfono. No es específico de un país o región, sin embargo, en este momento solo se detectan números de teléfono de EE. UU. y Canadá.

Tipo de datos	Descripción
USA_ATIN	El número de identificación fiscal de adopción de EE. UU. emitido por el Servicio de Impuestos Internos.
USA_CPT_CODE	El código CPT (específico para EE. UU.).
USA_DEA_NUMBER	El número DEA (específico para EE. UU.).
USA_DRIVING_LICENSE	El número de licencia de conducir (específico para EE. UU).
USA_HCPCS_CODE	El código HCPCS (específico para EE. UU.).
USA_HEALTH_INSURANCE_CLAIM_NUMBER	Número de reclamación del seguro médico (específico para EE. UU.).
USA_ITIN	El ITIN (para ciudadanos o entidades de EE. UU.).
USA_MEDICARE_BENEFICIARY_IDENTIFIER	Identificador de beneficiario de Medicare (específico para EE. UU.).
USA_NATIONAL_DRUG_CODE	El código NDC (específico para EE. UU.).
USA_NATIONAL_PROVIDER_IDENTIFIER	El número de identificación nacional del proveedor (específico para EE. UU.).
USA_PASSPORT_NUMBER	El número de pasaporte (para ciudadanos estadounidenses).
USA_PTIN	El número de identificación fiscal del preparado r de impuestos estadounidense emitido por el Servicio de Impuestos Internos.
USA_SSN	El número de seguro social (para ciudadanos estadounidenses).

Tipos de datos de Argentina

Tipo de datos	Descripción
ARGENTINA_TAX_IDENTIFICATIO N_NUMBER	Número de identificación fiscal de Argentina. También conocido como CUIT o CUIL.

Tipos de datos australianos

Tipo de datos	Descripción
AUSTRALIA_BUSINESS_NUMBER	Número comercial de Australia (ABN). Un identificador único emitido por el Registro Comercial de Australia (ABR) para identificar a las empresas ante el Gobierno y la comunidad.
AUSTRALIA_COMPANY_NUMBER	Número de empresa de Australia (ACN). Identificador único emitido por la Comisión Australiana de Valores e Inversiones.
AUSTRALIA_DRIVING_LICENSE	Un número de licencia para conductores de Australia.
AUSTRALIA_MEDICARE_NUMBER	Número de Medicare australiano. Identificador personal emitido por la Comisión de Seguros de Salud australiano.
AUSTRALIA_PASSPORT_NUMBER	Número de pasaporte australiano.
AUSTRALIA_TAX_FILE_NUMBER	Número de registro fiscal de Australia (TFN). Emitido por la Oficina de Impuestos de Australia (ATO) a los contribuyentes (individu os, empresas, etc.) para transacciones fiscales.

Tipos de datos de Austria

Tipo de datos	Descripción
AUSTRIA_DRIVING_LICENSE	El número de licencia de conducir (específico para Austria).
AUSTRIA_PASSPORT_NUMBER	El número de pasaporte (específico para Austria).
AUSTRIA_SSN	El número de seguro social (para ciudadanos de Austria).
AUSTRIA_TAX_IDENTIFICATION_NUMBER	Número de identificación fiscal (específico para Austria).
AUSTRIA_VALUE_ADDED_TAX	Impuesto al valor agregado (específico para Austria).

Tipos de datos de los Balcanes

Tipo de datos	Descripción
BOSNIA_UNIQUE_MASTER_CITIZEN_NUMBER	Número único de ciudadano maestro (JMBG) para los ciudadanos de Bosnia-Herzegovina.
KOSOVO_UNIQUE_MASTER_CITIZEN_NUMBER	Número único de ciudadano maestro (JMBG) de Kosovo.
MACEDONIA_UNIQUE_MASTER_CITIZEN_NUMBER	Número único de ciudadano mayor de Macedonia.
MONTENEGRO_UNIQUE_MASTER_CITIZEN_NUMBER	Número único de ciudadano maestro (JMBG) de Montenegro.
SERBIA_UNIQUE_MASTER_CITIZEN_NUMBER	Número único de ciudadano maestro (JMBG) de Serbia.
SERBIA_VALUE_ADDED_TAX	Impuesto sobre el valor agregado (específico para Serbia).

Tipo de datos	Descripción
VOJVODINA_UNIQUE_MASTER_CITIZEN_NUMBER	Número único de ciudadano maestro (JMBG) para Voivodina.

Tipos de datos de Bélgica

Tipo de datos	Descripción
BELGIUM_DRIVING_LICENSE	El número de licencia de conducir (específico para Bélgica).
BELGIUM_NATIONAL_IDENTIFICATION_NUMBER	El número nacional belga (BNN).
BELGIUM_PASSPORT_NUMBER	El número de pasaporte (específico para Bélgica).
BELGIUM_TAX_IDENTIFICATION_NUMBER	Número de identificación fiscal (específico para Bélgica).
BELGIUM_VALUE_ADDED_TAX	Impuesto sobre el valor agregado (específico para Bélgica).

Tipos de datos de Brasil

Tipo de datos	Descripción
BRAZIL_BANK_ACCOUNT	El número de cuenta bancaria (específico para Brasil).
BRAZIL_NATIONAL_IDENTIFICATION_NUMBER	El identificador nacional (específico para Brasil).
BRAZIL_NATIONAL_REGISTRY_OF_LEGAL_ENTITIES_NUMBER	El número de identificación emitido a las empresas (específico de Brasil), también conocido como CNPJ.

Tipo de datos	Descripción
BRAZIL_NATURAL_PERSON_REGISTRY_NUMBER	Número de registro de persona física, también conocido como CPF.

Tipos de datos de Bulgaria

Tipo de datos	Descripción
BULGARIA_DRIVING_LICENSE	El número de licencia de conducir (específico para Bulgaria).
BULGARIA_UNIFORM_CIVIL_NUMBER	Número civil unificado (EGN) que sirve como número de identificación nacional.
BULGARIA_VALUE_ADDED_TAX	Impuesto sobre el valor agregado (específico para Bulgaria).

Tipos de datos de Canadá

Tipo de datos	Descripción
CANADA_DRIVING_LICENSE	El número de licencia de conducir (específico para Canadá).
CANADA_GOVERNMENT_IDENTIFICATION_CARD_NUMBER	El identificador nacional (específico para Canadá).
CANADA_PASSPORT_NUMBER	El número de pasaporte (específico para Canadá).
CANADA_PERMANENT_RESIDENCE_NUMBER	Número de residencia permanente (número de tarjeta PR).
CANADA_PERSONAL_HEALTH_NUMBER	El identificador único de la asistencia sanitaria (número PHN).

Tipo de datos	Descripción
CANADA_SOCIAL_INSURANCE_NUMBER	El número de seguro social (SIN) en Canadá.

Tipos de datos de Chile

Tipo de datos	Descripción
CHILE_DRIVING_LICENSE	El número de licencia de conducir (específico para Chile).
CHILE_NATIONAL_IDENTIFICATION_NUMBER	El identificador nacional de Chile, también conocido como RUT o RUN.

Tipos de datos de China, Hong Kong, Macao y Taiwán

Tipo de datos	Descripción
CHINA_IDENTIFICATION	El identificador de China.
CHINA_LICENSE_PLATE_NUMBER	El número de licencia de conducir (específico para China).
CHINA_MAINLAND_TRAVEL_PERMIT_ID_HONG_KONG_MACAU	El permiso de viaje a China continental para residentes de Hong Kong y Macao.
CHINA_MAINLAND_TRAVEL_PERMIT_ID_TAIWAN	El permiso de viaje a China continental para residentes de Taiwán emitido por el Gobierno de la República Popular China (PRC).
CHINA_PASSPORT_NUMBER	El número de pasaporte (específico para China).
CHINA_PHONE_NUMBER	El número de teléfono (específico para China)
HONG_KONG_IDENTITY_CARD	El documento de identidad oficial emitido por el Departamento de Inmigración de Hong Kong.

Tipo de datos	Descripción
MACAU_RESIDENT_IDENTITY_CARD	La tarjeta de identidad de residente de Macao o BIR es una tarjeta de identidad oficial emitida por la Oficina de Servicios de Identificación de Macao.
TAIWAN_NATIONAL_IDENTIFICATION_NUMBER	El identificador nacional (específico para Taiwán).
TAIWAN_PASSPORT_NUMBER	El número de pasaporte (específico para Taiwán).

Tipos de datos de Colombia

Tipo de datos	Descripción
COLOMBIA_PERSONAL_IDENTIFICATION_NUMBER	Identificador único asignado a los colombianos al nacer.
COLOMBIA_TAX_IDENTIFICATION_NUMBER	Número de identificación fiscal (específico para Colombia).

Tipos de datos de Croacia

Tipo de datos	Descripción
CROATIA_DRIVING_LICENSE	El número de licencia de conducir (específico para Croacia).
CROATIA_IDENTITY_NUMBER	El identificador nacional (específico para Croacia).
CROATIA_PASSPORT_NUMBER	El número de pasaporte (específico para Croacia).

Tipo de datos	Descripción
CROATIA_PERSONAL_IDENTIFICATION_NUMBER	El número de identificación personal (OIB).

Tipos de datos de Chipre

Tipo de datos	Descripción
CYPRUS_DRIVING_LICENSE	El número de licencia de conducir (específico para Cyprus).
CYPRUS_NATIONAL_IDENTIFICATION_NUMBER	El carné de identidad chipriota.
CYPRUS_PASSPORT_NUMBER	El número de pasaporte (específico para Chipre).
CYPRUS_TAX_IDENTIFICATION_NUMBER	Número de identificación fiscal (específico para Chipre).
CYPRUS_VALUE_ADDED_TAX	Impuesto sobre el valor agregado (específico para Chipre).

Tipos de datos de Chequia

Tipo de datos	Descripción
CZECHIA_DRIVING_LICENSE	El número de licencia de conducir (específico para Chequia).
CZECHIA_PERSONAL_IDENTIFICATION_NUMBER	El número de identificación personal (específico para Chequia).
CZECHIA_VALUE_ADDED_TAX	Impuesto sobre el valor agregado (específico para Chequia).

Tipos de datos de Dinamarca

Tipo de datos	Descripción
DENMARK_DRIVING_LICENSE	El número de licencia de conducir (específico para Dinamarca).
DENMARK_PERSONAL_IDENTIFICATION_NUMBER	El número de identificación personal (específico para Dinamarca).
DENMARK_TAX_IDENTIFICATION_NUMBER	Número de identificación fiscal (específico para Dinamarca).
DENMARK_VALUE_ADDED_TAX	Impuesto sobre el valor agregado (específico para Dinamarca).

Tipos de datos de Estonia

Tipo de datos	Descripción
ESTONIA_DRIVING_LICENSE	El número de licencia de conducir (específico para Estonia).
ESTONIA_PASSPORT_NUMBER	El número de pasaporte (específico para Estonia).
ESTONIA_PERSONAL_IDENTIFICATION_CODE	El número de identificación personal (específico para Estonia).
ESTONIA_VALUE_ADDED_TAX	Impuesto sobre el valor agregado (específico para Estonia).

Tipos de datos de Finlandia

Tipo de datos	Descripción
FINLAND_DRIVING_LICENSE	El número de licencia de conducir (específico para Finlandia).
FINLAND_HEALTH_INSURANCE_NUMBER	El número de seguro médico (específico para Finlandia).
FINLAND_NATIONAL_IDENTIFICATION_NUMBER	El número de identificación nacional (específico para Finlandia).
FINLAND_PASSPORT_NUMBER	El número de pasaporte (específico para Finlandia).
FINLAND_VALUE_ADDED_TAX	Impuesto sobre el valor agregado (específico para Finlandia).

Tipos de datos de Francia

Tipo de datos	Descripción
FRANCE_BANK_ACCOUNT	El número de cuenta bancaria (específico para Francia).
FRANCE_DRIVING_LICENSE	El número de licencia de conducir (específico para Francia).
FRANCE_HEALTH_INSURANCE_NUMBER	Número de seguro médico de Francia.
FRANCE_INSEE_CODE	Número de seguridad social, SSN o NIR de Francia.
FRANCE_NATIONAL_IDENTIFICATION_NUMBER	Número de identificación nacional (CNI) de Francia.
FRANCE_PASSPORT_NUMBER	El número de pasaporte (específico para Francia).

Tipo de datos	Descripción
FRANCE_TAX_IDENTIFICATION_NUMBER	Número de identificación fiscal (específico para Francia).
FRANCE_VALUE_ADDED_TAX	Impuesto sobre el valor agregado (específico para Francia).

Tipos de datos de Alemania

Tipo de datos	Descripción
GERMANY_BANK_ACCOUNT	El número de cuenta bancaria (específico para Alemania)
GERMANY_DRIVING_LICENSE	El número de licencia de conducir (específico para Alemania).
GERMANY_PASSPORT_NUMBER	El número de pasaporte (específico para Alemania).
GERMANY_PERSONAL_IDENTIFICATION_NUMBER	El número de identificación personal (específico para Alemania).
GERMANY_TAX_IDENTIFICATION_NUMBER	Número de identificación fiscal (específico para Alemania).
GERMANY_VALUE_ADDED_TAX	Impuesto sobre el valor agregado (específico para Alemania).

Tipos de datos de Grecia

Tipo de datos	Descripción
GREECE_DRIVING_LICENSE	El número de licencia de conducir (específico para Grecia).

Tipo de datos	Descripción
GREECE_PASSPORT_NUMBER	El número de pasaporte (específico para Grecia).
GREECE_SSN	El número de seguro social (para ciudadanos de Grecia).
GREECE_TAX_IDENTIFICATION_NUMBER	Número de identificación fiscal (específico para Grecia).
GREECE_VALUE_ADDED_TAX	Impuesto sobre el valor agregado (específico para Grecia).

Tipos de datos de Hungría

Tipo de datos	Descripción
HUNGARY_DRIVING_LICENSE	El número de licencia de conducir (específico para Hungría).
HUNGARY_PASSPORT_NUMBER	El número de pasaporte (específico para Hungría).
HUNGARY_SSN	El número de seguro social (para ciudadanos húngaros).
HUNGARY_TAX_IDENTIFICATION_NUMBER	Número de identificación fiscal (específico para Hungría).
HUNGARY_VALUE_ADDED_TAX	Impuesto sobre el valor agregado (específico para Hungría).

Tipos de datos de Islandia

Tipo de datos	Descripción
ICELAND_NATIONAL_IDENTIFICATION_NUMBER	El identificador nacional (específico para Islandia).
ICELAND_PASSPORT_NUMBER	El número de pasaporte (específico para Islandia).
ICELAND_VALUE_ADDED_TAX	Impuesto sobre el valor agregado (específico para Islandia).

Tipos de datos de India

Tipo de datos	Descripción
INDIA_AADHAAR_NUMBER	Número de identificación de Aadhaar emitido por la Autoridad de Identificación Única de la India.
INDIA_PERMANENT_ACCOUNT_NUMBER	Número de cuenta permanente de la India (PAN).

Tipos de datos de Indonesia

Tipo de datos	Descripción
INDONESIA_IDENTITY_CARD_NUMBER	El identificador nacional (específico para Indonesia).

Tipos de datos de Irlanda

Tipo de datos	Descripción
IRELAND_DRIVING_LICENSE	El número de licencia de conducir (específico para Irlanda).

Tipo de datos	Descripción
IRELAND_PASSPORT_NUMBER	El número de pasaporte (específico para Irlanda).
IRELAND_PERSONAL_PUBLIC_SERVICE_NUMBER	Número de servicio público personal (PPS) de Irlanda.
IRELAND_TAX_IDENTIFICATION_NUMBER	Número de identificación fiscal (específico para Irlanda).
IRELAND_VALUE_ADDED_TAX	Impuesto sobre el valor agregado (específico para Irlanda).

Tipos de datos de Israel

Tipo de datos	Descripción
ISRAEL_IDENTIFICATION_NUMBER	El identificador nacional (específico para Israel).

Tipos de datos de Italia

Tipo de datos	Descripción
ITALY_BANK_ACCOUNT	El número de cuenta bancaria (específico para Italia).
ITALY_DRIVING_LICENSE	El número de licencia de conducir (específico para Italia).
ITALY_FISCAL_CODE	El número identificador, también conocido como Codice Fiscale italiano.
ITALY_PASSPORT_NUMBER	El número de pasaporte (específico para Italia).

Tipo de datos	Descripción
ITALY_VALUE_ADDED_TAX	Impuesto sobre el valor agregado (específico para Italia).

Tipos de datos de Japón

Tipo de datos	Descripción
JAPAN_BANK_ACCOUNT	Cuenta bancaria en Japón.
JAPAN_DRIVING_LICENSE	Un número de licencia de conducir de Japón.
JAPAN_MY_NUMBER	El identificador único para los ciudadanos o corporaciones de Japón utilizado para la administración tributaria, la administración de la seguridad social y la respuesta ante desastres
JAPAN_PASSPORT_NUMBER	Número de pasaporte de Japón.

Tipos de datos de Corea

Tipo de datos	Descripción
KOREA_PASSPORT_NUMBER	El número de pasaporte (específico para Corea).
KOREA_RESIDENCE_REGISTRATION_NUMBER_FOR_CITIZENS	Número de registro de residencia en Corea para residentes.
KOREA_RESIDENCE_REGISTRATION_NUMBER_FOR_FOREIGNERS	Número de registro de residencia en Corea para extranjeros.

Tipos de datos de Letonia

Tipo de datos	Descripción
LATVIA_DRIVING_LICENSE	El número de licencia de conducir (específico para Letonia).
LATVIA_PASSPORT_NUMBER	El número de pasaporte (específico para Letonia).
LATVIA_PERSONAL_IDENTIFICATION_NUMBER	El número de identificación personal (específico para Letonia).
LATVIA_VALUE_ADDED_TAX	Impuesto sobre el valor agregado (específico para Letonia).

Tipos de datos de Liechtenstein

Tipo de datos	Descripción
LIECHTENSTEIN_NATIONAL_IDENTIFICATION_NUMBER	El identificador nacional (específico para Liechtenstein).
LIECHTENSTEIN_PASSPORT_NUMBER	El número de pasaporte (específico para Liechtenstein).
LIECHTENSTEIN_TAX_IDENTIFICATION_NUMBER	Número de identificación fiscal (específico para Liechtenstein).

Tipos de datos de Lituania

Tipo de datos	Descripción
LITUANIA_DRIVING_LICENSE	El número de licencia de conducir (específico para Lituania).
LITHUANIA_PERSONAL_IDENTIFICATION_NUMBER	El número de identificación personal (específico para Lituania).

Tipo de datos	Descripción
LITUANIA_TAX_IDENTIFICATION_NUMBER	Número de identificación fiscal (específico para Lituania).
LITHUANIA_VALUE_ADDED_TAX	Impuesto sobre el valor agregado (específico para Lituania).

Tipos de datos de Luxemburgo

Tipo de datos	Descripción
LUXEMBOURG_DRIVING_LICENSE	El número de licencia de conducir (específico para Luxemburgo).
LUXEMBOURG_NATIONAL_INDIVIDUAL_NUMBER	El identificador nacional (específico para Luxemburgo).
LUXEMBOURG_PASSPORT_NUMBER	El número de pasaporte (específico para Luxemburgo).
LUXEMBOURG_TAX_IDENTIFICATION_NUMBER	Número de identificación fiscal (específico para Luxemburgo).
LUXEMBOURG_VALUE_ADDED_TAX	Impuesto sobre el valor agregado (específico para Luxemburgo).

Tipos de datos de Malasia

Tipo de datos	Descripción
MALASIA_MYKAD_NUMBER	El identificador nacional (específico para Malasia).
MALAYSIA_PASSPORT_NUMBER	El número de pasaporte (específico para Malasia).

Tipos de datos de Malta

Tipo de datos	Descripción
MALTA_DRIVING_LICENSE	El número de licencia de conducir (específico para Malta).
MALTA_NATIONAL_IDENTIFICATION_NUMBER	El identificador nacional (específico para Malta).
MALTA_TAX_IDENTIFICATION_NUMBER	Número de identificación fiscal (específico para Malta).
MALTA_VALUE_ADDED_TAX	Impuesto sobre el valor agregado (específico para Malta).

Tipos de datos de México

Tipo de datos	Descripción
MEXICO_CLABE_NUMBER	Número de banco (CLABE (Clave Bancaria Estandarizada) de México).
MEXICO_DRIVING_LICENSE	El número de licencia de conducir (específico para México).
MEXICO_PASSPORT_NUMBER	El número de pasaporte (específico para México).
MEXICO_TAX_IDENTIFICATION_NUMBER	Número de identificación fiscal (específico para México).
MEXICO_UNIQUE_POPULATION_REGISTRY_CODE	La Clave Única de Registro de Población (CURP) es el código de identidad único de México.

Tipos de datos de los Países Bajos

Tipo de datos	Descripción
NETHERLANDS_CITIZEN_SERVICE_NUMBER	Número de ciudadano neerlandés (BSN, burgerservicenummer).
NETHERLANDS_DRIVING_LICENSE	El número de licencia de conducir (específico para los Países Bajos).
NETHERLANDS_PASSPORT_NUMBER	El número de pasaporte (específico para los Países Bajos).
NETHERLANDS_TAX_IDENTIFICATION_NUMBER	Número de identificación fiscal (específico para los Países Bajos).
NETHERLANDS_VALUE_ADDED_TAX	Impuesto sobre el valor agregado (específico para los Países Bajos).
NETHERLANDS_BANK_ACCOUNT	El número de cuenta bancaria (específico para los Países Bajos)

Tipos de datos de Nueva Zelanda

Tipo de datos	Descripción
NEW_ZEALAND_DRIVING_LICENSE	El número de licencia de conducir (específico para Nueva Zelanda).
NEW_ZEALAND_NATIONAL_HEALTH_INDEX_NUMBER	Número del índice nacional de salud de Nueva Zelanda.
NEW_ZEALAND_TAX_IDENTIFICATION_NUMBER	Número de identificación fiscal, también conocido como número de impuestos internos (específico para Nueva Zelanda).

Tipos de datos de Noruega

Tipo de datos	Descripción
NORWAY_BIRTH_NUMBER	Número de identidad nacional noruego.
NORWAY_DRIVING_LICENSE	El número de licencia de conducir (específico para Noruega).
NORWAY_HEALTH_INSURANCE_NUMBER	Número de seguro médico noruego.
NORWAY_NATIONAL_IDENTIFICATION_NUMBER	El número de identificación nacional (específico para Noruega).
NORWAY_VALUE_ADDED_TAX	Impuesto sobre el valor agregado (específico para Noruega).

Tipos de datos de Filipinas

Tipo de datos	Descripción
PHILIPPINES_DRIVING_LICENSE	El número de licencia de conducir (específico para Filipinas).
PHILIPPINES_PASSPORT_NUMBER	El número de pasaporte (específico para Filipinas).

Tipos de datos de Polonia

Tipo de datos	Descripción
POLAND_DRIVING_LICENSE	El número de licencia de conducir (específico para Polonia).
POLAND_IDENTIFICATION_NUMBER	El identificador del Polonia.
POLAND_PASSPORT_NUMBER	El número de pasaporte (específico para Polonia).

Tipo de datos	Descripción
POLAND_REGON_NUMBER	El número identificador de REGON, también conocido como número de identificación estadística.
POLAND_SSN	El número de seguro social (para ciudadanos de Polonia).
POLAND_TAX_IDENTIFICATION_NUMBER	Número de identificación fiscal (específico para Polonia).
POLAND_VALUE_ADDED_TAX	Impuesto sobre el valor agregado (específico para Polonia).

Tipos de datos de Portugal

Tipo de datos	Descripción
PORTUGAL_DRIVING_LICENSE	El número de licencia de conducir (específico para Portugal).
PORTUGAL_NATIONAL_IDENTIFICATION_NUMBER	El número de identificación nacional del proveedor (específico para Portugal).
PORTUGAL_PASSPORT_NUMBER	El número de pasaporte (específico para Portugal).
PORTUGAL_TAX_IDENTIFICATION_NUMBER	Número de identificación fiscal (específico para Portugal).
PORTUGAL_VALUE_ADDED_TAX	Impuesto sobre el valor agregado (específico para Portugal).

Tipos de datos de Rumanía

Tipo de datos	Descripción
ROMANIA_DRIVING_LICENSE	El número de licencia de conducir (específico para Rumanía).
ROMANIA_NUMERICAL_PERSONAL_CODE	El número de identificación personal (específico para Rumanía).
ROMANIA_PASSPORT_NUMBER	El número de pasaporte (específico para Rumanía).
ROMANIA_VALUE_ADDED_TAX	Impuesto sobre el valor agregado (específico para Rumanía).

Tipos de datos de Singapur

Tipo de datos	Descripción
SINGAPORE_DRIVING_LICENSE	El número de licencia de conducir (específico para Singapur).
SINGAPORE_NATIONAL_REGISTRY_IDENTIFICATION_NUMBER	El carné de identidad de registro nacional de Singapur.
SINGAPORE_PASSPORT_NUMBER	El número de pasaporte (específico para Singapur).
SINGAPORE_UNIQUE_ENTITY_NUMBER	Número de entidad único (UEN) de Singapur

Tipos de datos de Eslovaquia

Tipo de datos	Descripción
SLOVAKIA_DRIVING_LICENSE	El número de licencia de conducir (específico para Eslovaquia).

Tipo de datos	Descripción
SLOVAKIA_NATIONAL_IDENTIFICATION_NUMBER	El número de identificación nacional (específico para Eslovaquia).
SLOVAKIA_PASSPORT_NUMBER	El número de pasaporte (específico para Eslovaquia).
SLOVAKIA_VALUE_ADDED_TAX	Impuesto sobre el valor agregado (específico para Eslovaquia).

Tipos de datos de Eslovenia

Tipo de datos	Descripción
SLOVENIA_DRIVING_LICENSE	El número de licencia de conducir (específico para Eslovenia).
SLOVENIA_PASSPORT_NUMBER	El número de pasaporte (específico para Eslovenia).
ESLOVENIA_TAX_IDENTIFICATION_NUMBER	Número de identificación fiscal (específico para Eslovenia).
SLOVENIA_UNIQUE_MASTER_CITIZEN_NUMBER	Número de ciudadano maestro (JMBG) único para los ciudadanos de Eslovenia.
SLOVENIA_VALUE_ADDED_TAX	Impuesto sobre el valor agregado (específico para Eslovenia).

Tipos de datos de Sudáfrica

Tipo de datos	Descripción
SOUTH_AFRICA_PERSONAL_IDENTIFICATION_NUMBER	El número de identificación personal (específico para Sudáfrica).

Tipos de datos de España

Tipo de datos	Descripción
SPAIN_BANK_ACCOUNT	El número de cuenta bancaria (específico para España).
SPAIN_DNI	El documento nacional de identidad (Documento Nacional de Identidad) de España.
SPAIN_DRIVING_LICENSE	El número de licencia de conducir (específico para España).
SPAIN_NIE	El número de identidad del extranjero (específico para España), también conocido como NIE.
SPAIN_NIF	Número de identificación fiscal (específico para España), también conocido como NIF.
SPAIN_PASSPORT_NUMBER	El número de pasaporte (específico para España).
SPAIN_SSN	El número de seguro social (para ciudadanos de España).
SPAIN_VALUE_ADDED_TAX	Impuesto sobre el valor agregado (específico para España).

Tipos de datos de Sri Lanka

Tipo de datos	Descripción
SRI_LANKA_NATIONAL_IDENTIFICATION_NUMBER	El identificador nacional (específico para Sri Lanka).

Tipos de datos de Suecia

Tipo de datos	Descripción
SWEDEN_DRIVING_LICENSE	El número de licencia de conducir (específico para Suecia).
SWEDEN_PASSPORT_NUMBER	El número de pasaporte (específico para Suecia).
SWEDEN_PERSONAL_IDENTIFICATION_NUMBER	El número de identificación nacional (específico para Suecia).
SWEDEN_TAX_IDENTIFICATION_NUMBER	Número de identificación fiscal de Suecia (personnummer).
SWEDEN_VALUE_ADDED_TAX	Impuesto sobre el valor agregado (específico para Suecia).

Tipos de datos de Suiza

Tipo de datos	Descripción
SWITZERLAND_AHV	El número de seguro social para ciudadanos suizos (AHV).
SWITZERLAND_HEALTH_INSURANCE_NUMBER	Número de seguro médico suizo.
SWITZERLAND_PASSPORT_NUMBER	El número de pasaporte (específico para Suiza).
SWITZERLAND_VALUE_ADDED_TAX	Impuesto sobre el valor agregado (específico para Suiza).

Tipos de datos de Tailandia

Tipo de datos	Descripción
THAILAND_PASSPORT_NUMBER	El número de pasaporte (específico para Tailandia).
THAILAND_PERSONAL_IDENTIFICATION_NUMBER	El número de identificación personal (específico para Tailandia).

Tipos de datos de Turquía

Tipo de datos	Descripción
TURKEY_NATIONAL_IDENTIFICATION_NUMBER	El número de identificación nacional (específico para Turquía).
TURKEY_PASSPORT_NUMBER	El número de pasaporte (específico para Turquía).
TURKEY_VALUE_ADDED_TAX	Impuesto sobre el valor agregado (específico para Turquía).

Tipos de datos de Ucrania

Tipo de datos	Descripción
UKRAINE_INDIVIDUAL_IDENTIFICATION_NUMBER	El identificador único (específico para Ucrania).
UKRAINE_PASSPORT_NUMBER_DOMESTIC	El número de pasaporte nacional (específico para Ucrania).
UKRAINE_PASSPORT_NUMBER_INTERNATIONAL	El número de pasaporte internacional (específico para Ucrania).

Tipos de datos de los Emiratos Árabes Unidos (UAE)

Tipo de datos	Descripción
UNITED_ARAB_EMIRATES_PERSONAL_NUMBER	El número de identificación personal (específico para los Emiratos Árabes Unidos).

Tipos de datos para Reino Unido

Tipo de datos	Descripción
UK_BANK_ACCOUNT	Cuenta bancaria en el Reino Unido (UK).
UK_BANK_SORT_CODE	Código de clasificación bancaria del Reino Unido (UK). Los códigos de clasificación son códigos bancarios que se utilizan para enrutar transferencias de dinero entre bancos de sus respectivos países a través de sus respectivas organizaciones de liquidación.
UK_DRIVING_LICENSE	El número de licencia de conducir del Reino Unido de Gran Bretaña e Irlanda del Norte (específico del Reino Unido)
UK_ELECTORAL_ROLL_NUMBER	El número de registro electoral (ERN) es el número de identificación que se expide a una persona para el registro electoral del Reino Unido. El formato de este número está especificado por las normas gubernamentales del Reino Unido de la Oficina del Gabinete del Reino Unido.
UK_NATIONAL_HEALTH_SERVICE_NUMBER	El número del Servicio Nacional de Salud (National Health Service, NHS) es el número único asignado a un usuario registrado de los servicios de salud pública en el Reino Unido.
UK_NATIONAL_INSURANCE_NUMBER	El número de seguro nacional (National Insurance Number, NINO) es un número que

Tipo de datos	Descripción
	se utiliza en el Reino Unido (UK) con el objetivo de identificar a una persona para el programa de seguro nacional o el sistema de seguridad social. Este número es lo que a veces se denomina NI NO o NINO.
UK_PASSPORT_NUMBER	Número de pasaporte del Reino Unido (UK).
UK_UNIQUE_TAXADOR_REFERENCE_NUMBER	El número de referencia tributaria única (UTR) del Reino Unido (Reino Unido). Un identificador utilizado por el Gobierno del Reino Unido para gestionar el sistema tributario.
UK_VALUE_ADDED_TAX	El IVA es un impuesto al consumo que corre a cargo del consumidor final. El IVA se paga por cada transacción del proceso de fabricación y distribución. Para el Reino Unido, el número de IVA lo emite la oficina de IVA de la región en la que está establecida la empresa.
UK_PHONE_NUMBER	Número de teléfono del Reino Unido (UK).

Tipos de datos de Venezuela

Tipo de datos	Descripción
VENEZUELA_DRIVING_LICENSE	El número de licencia de conducir (específico para Venezuela).
VENEZUELA_NATIONAL_IDENTIFICATION_NUMBER	El número de identificación nacional (específico para Venezuela).
VENEZUELA_VALUE_ADDED_TAX	Impuesto al valor agregado (específico para Venezuela).

Uso de una detección de datos confidenciales detallada

> **Note**
>
> Las acciones detalladas solo están disponibles en las versiones de AWS Glue 3.0 y 4.0. Esto incluye la experiencia AWS Glue Studio. Los cambios persistentes en el registro de auditoría tampoco están disponibles en la versión 2.0.
> En todos los trabajos visuales de las versiones de AWS Glue Studio 3.0 y 4.0 se creará un script que utilizará automáticamente API de acciones detalladas.

La transformación Detectar datos confidenciales proporciona la capacidad de detectar, enmascarar o eliminar entidades definidas por el usuario o predefinidas por AWS Glue. Las acciones detalladas permiten además aplicar una acción específica por entidad. Los beneficios adicionales incluyen:

- Rendimiento mejorado, ya que las acciones se aplican en cuanto se detectan los datos.
- La opción de incluir o excluir columnas específicas.
- La capacidad de utilizar un enmascaramiento parcial. Esto le permite enmascarar parcialmente las entidades de datos confidenciales detectadas, en lugar de enmascarar toda la cadena. Se admiten ambos parámetros simples con compensaciones y expresiones regulares.

A continuación, se muestran fragmentos de código de las API de detección de datos confidenciales y de las acciones detalladas que se utilizan en los trabajos de muestra a los que se hace referencia en la siguiente sección.

API de detección: las acciones detalladas utilizan el nuevo parámetro `detectionParameters`:

```
def detect(
    frame: DynamicFrame,
    detectionParameters: JsonOptions,
    outputColumnName: String = "DetectedEntities",
    detectionSensitivity: String = "LOW"
): DynamicFrame = {}
```

Uso de las API de detección de datos confidenciales con acciones detalladas

Las API de detección de datos confidenciales que utilizan detect analizan los datos proporcionados, determinan si las filas o columnas son tipos de entidades de datos confidenciales y ejecutarán las acciones especificadas por el usuario para cada tipo de entidad.

Uso de la API de detección con acciones detalladas

Use la API de detección y especifique el `outputColumnName` y los `detectionParameters`.

```
   object GlueApp {
     def main(sysArgs: Array[String]) {

       val spark: SparkContext = new SparkContext()
       val glueContext: GlueContext = new GlueContext(spark)

       // @params: [JOB_NAME]
       val args = GlueArgParser.getResolvedOptions(sysArgs, Seq("JOB_NAME").toArray)
       Job.init(args("JOB_NAME"), glueContext, args.asJava)

       // Script generated for node S3 bucket. Creates DataFrame from data stored in
S3.
       val S3bucket_node1 =
glueContext.getSourceWithFormat(formatOptions=JsonOptions("""{"quoteChar":
"\"", "withHeader": true, "separator": ",", "optimizePerformance": false}"""),
connectionType="s3", format="csv", options=JsonOptions("""{"paths":
["s3://189657479688-ddevansh-pii-test-bucket/tiny_pii.csv"], "recurse": true}"""),
transformationContext="S3bucket_node1").getDynamicFrame()

       // Script generated for node Detect Sensitive Data. Will run detect API for the
DataFrame
       // detectionParameter contains information on which EntityType are being
detected
       // and what actions are being applied to them when detected.
       val DetectSensitiveData_node2 = EntityDetector.detect(
           frame = S3bucket_node1,
           detectionParameters = JsonOptions(
            """
               {
                   "PHONE_NUMBER": [
                       {
                           "action": "PARTIAL_REDACT",
                           "actionOptions": {
```

```
                                "numLeftCharsToExclude": "3",
                                "numRightCharsToExclude": "4",
                                "redactChar": "#"
                            },
                            "sourceColumnsToExclude": [ "Passport No", "DL NO#" ]
                        }
                    ],
                    "USA_PASSPORT_NUMBER": [
                        {
                            "action": "SHA256_HASH",
                            "sourceColumns": [ "Passport No" ]
                        }
                    ],
                    "USA_DRIVING_LICENSE": [
                        {
                            "action": "REDACT",
                            "actionOptions": {
                                "redactText": "USA_DL"
                            },
                            "sourceColumns": [ "DL NO#" ]
                        }
                    ]

                }
            """
            ),
            outputColumnName = "DetectedEntities"
        )

        // Script generated for node S3 bucket. Store Results of detect to S3 location
        val S3bucket_node3 = glueContext.getSinkWithFormat(connectionType="s3",
 options=JsonOptions("""{"path": "s3://189657479688-ddevansh-pii-test-bucket/
test-output/", "partitionKeys": []}"""), transformationContext="S3bucket_node3",
 format="json").writeDynamicFrame(DetectSensitiveData_node2)

        Job.commit()
      }
```

El script anterior creará un DataFrame desde una ubicación en Amazon S3 y, a continuación, ejecutará la API `detect`. Como la API `detect` requiere que el campo `detectionParameters` (un mapa del nombre de la entidad con una lista de todos los ajustes de acción que se van a utilizar en

esa entidad) esté representado por el objeto `JsonOptions` de AWS Glue, también nos permitirá ampliar la funcionalidad de la API.

Para cada acción especificada por entidad, introduzca una lista de todos los nombres de columna a los que quiere aplicar la combinación de entidad y acción. Esto le permite personalizar las entidades para detectarlas en cada columna de su conjunto de datos y omitir las entidades que sabe que no están en una columna específica. Esto también permite que sus trabajos sean más eficaces al no realizar llamadas de detección innecesarias a esas entidades y le permite realizar acciones exclusivas para cada combinación de columnas y entidades.

Si analizamos más detenidamente el `detectionParameters`, en el trabajo de muestra hay tres tipos de entidades. Estos son `Phone Number`, `USA_PASSPORT_NUMBER` y `USA_DRIVING_LICENSE`. Para cada uno de estos tipos de entidades, AWS Glue ejecutará diferentes acciones que son `PARTIAL_REDACT`, `SHA256_HASH`, `REDACT` y `DETECT`. Cada uno de los tipos de entidad también tiene `sourceColumns` que aplicar o `sourceColumnsToExclude` si se detecta.

> **Note**
>
> Solo se puede usar una acción de edición in situ (`PARTIAL_REDACT`, `SHA256_HASH` o `REDACT`) por columna, pero la acción `DETECT` se puede usar con cualquiera de estas acciones.

El campo `detectionParameters` tiene el siguiente diseño:

```
ENTITY_NAME -> List[Actions]
{
 "ENTITY_NAME": [{
  Action, // required
  ColumnSpecs,
  ActionOptionsMap
    }],
    "ENTITY_NAME2": [{
  ...
    }]
}
```

Los tipos de `actions` y `actionOptions` se enumeran a continuación:

```
DETECT
{
    # Required
    "action": "DETECT",
    # Optional, depending on action chosen
    "actionOptions": {
        // There are no actionOptions for DETECT
    },
    # 1 of below required, both can also used
    "sourceColumns": [
        "COL_1", "COL_2", ..., "COL_N"
    ],
    "sourceColumnsToExclude": [
        "COL_5"
    ]
}

SHA256_HASH
{
    # Required
    "action": "SHA256_HASH",
    # Required or optional, depending on action chosen
    "actionOptions": {
        // There are no actionOptions for SHA256_HASH
    },

    # 1 of below required, both can also used
    "sourceColumns": [
        "COL_1", "COL_2", ..., "COL_N"
    ],
    "sourceColumnsToExclude": [
        "COL_5"
    ]
}

REDACT
{
    # Required
    "action": "REDACT",
    # Required or optional, depending on action chosen
    "actionOptions": {
        // The text that is being replaced
```

```
        "redactText": "USA_DL"
    },

    # 1 of below required, both can also used
    "sourceColumns": [
        "COL_1", "COL_2", ..., "COL_N"
    ],
    "sourceColumnsToExclude": [
        "COL_5"
    ]
}

PARTIAL_REDACT
{
    # Required
    "action": "PARTIAL_REDACT",
    # Required or optional, depending on action chosen
    "actionOptions": {
        // number of characters to not redact from the left side
        "numLeftCharsToExclude": "3",
        // number of characters to not redact from the right side
        "numRightCharsToExclude": "4",
        // the partial redact will be made with this redacted character
        "redactChar": "#",
        // regex pattern for partial redaction
        "matchPattern": "[0-9]"
    },

    # 1 of below required, both can also used
    "sourceColumns": [
        "COL_1", "COL_2", ..., "COL_N"
    ],
    "sourceColumnsToExclude": [
        "COL_5"
    ]
}
```

Una vez que se ejecuta el script, los resultados se envían a la ubicación de Amazon S3 indicada. Puede ver sus datos en Amazon S3, pero con los tipos de entidades seleccionados sensibilizados en función de la acción seleccionada. En ese caso, tendríamos filas que tendrían el siguiente aspecto:

```
{
```

```
    "Name": "Colby Schuster",
    "Address": "39041 Antonietta Vista, South Rodgerside, Nebraska 24151",
    "Car Owned": "Fiat",
    "Email": "Kitty46@gmail.com",
    "Company": "O'Reilly Group",
    "Job Title": "Dynamic Functionality Facilitator",
    "ITIN": "991-22-2906",
    "Username": "Cassandre.Kub43",
    "SSN": "914-22-2906",
    "DOB": "2020-08-27",
    "Phone Number": "1-2#######1718",
    "Bank Account No": "69741187",
    "Credit Card Number": "6441-6289-6867-2162-2711",
    "Passport No": "94f311e93a623c72ccb6fc46cf5f5b0265ccb42c517498a0f27fd4c43b47111e",
    "DL NO#": "USA_DL"
}
```

En el guion anterior, el `Phone Number` se redactó parcialmente con #. Se cambió el `Passport No` a un hash SHA256. Se detectó el `DL NO#` como un número de licencia de conducir de EE. UU. y estaba redactado como “USA_DL” tal y como aparecía en los `detectionParameters`.

> **Note**
>
> La API classifyColumns no está disponible para su uso con acciones específicas debido a la naturaleza de la API. Esta API realiza un muestreo de columnas (ajustable por el usuario, pero tiene valores predeterminados) para realizar la detección más rápidamente. Por este motivo, las acciones detalladas requieren iterar todos los valores.

Registro de auditoría persistente

Una nueva característica que se introdujo con acciones detalladas (pero que también está disponible cuando se utilizan las API normales) es la presencia de un registro de auditoría persistente. Actualmente, al ejecutar la API de detección se añade un parámetro de columna adicional (predeterminado en `DetectedEntities` pero se puede personalizar mediante `outputColumnName`) con metadatos de detección de PII. Ahora tiene una clave de metadatos “actionUsed”, que es una de las siguientes:`DETECT`, `PARTIAL_REDACT`, `SHA256_HASH`, `REDACT`.

```
"DetectedEntities": {
```

```
    "Credit Card Number": [
        {
            "entityType": "CREDIT_CARD",
            "actionUsed": "DETECT",
            "start": 0,
            "end": 19
        }
    ],
    "Phone Number": [
        {
            "entityType": "PHONE_NUMBER",
            "actionUsed": "REDACT",
            "start": 0,
            "end": 14
        }
    ]
}
```

Incluso los clientes que utilicen las API sin acciones detalladas, como por ejemplo `detect(entityTypesToDetect, outputColumnName)`, verán este registro de auditoría persistente en el marco de datos resultante.

Los clientes que utilicen API con acciones detalladas verán todas las acciones, independientemente de si están redactadas o no. Ejemplo:

```
+---------------------+---------------
+-------------------------------------------------------------------------------------------------------
+
| Credit Card Number  |  Phone Number  |
                                              DetectedEntities
                                                                    |
+---------------------+---------------
+-------------------------------------------------------------------------------------------------------
+
| 622126741306XXXX    | +12#####7890   | {"Credit Card Number":
[{"entityType":"CREDIT_CARD","actionUsed":"PARTIAL_REDACT","start":0,"end":16}],"Phone
 Number":
[{"entityType":"PHONE_NUMBER","actionUsed":"PARTIAL_REDACT","start":0,"end":12}]}} |
| 6221 2674 1306 XXXX | +12#######7890 | {"Credit Card Number":
[{"entityType":"CREDIT_CARD","actionUsed":"PARTIAL_REDACT","start":0,"end":19}],"Phone
 Number":
[{"entityType":"PHONE_NUMBER","actionUsed":"PARTIAL_REDACT","start":0,"end":14}]}} |
```

```
| 6221-2674-1306-XXXX | 22#######7890  | {"Credit Card Number":
[{"entityType":"CREDIT_CARD","actionUsed":"PARTIAL_REDACT","start":0,"end":19}],"Phone
 Number":
[{"entityType":"PHONE_NUMBER","actionUsed":"PARTIAL_REDACT","start":0,"end":14}]}} |
+--------------------+---------------
+-------------------------------------------------------------------------------------------------------
+
```

Si no desea ver la columna DetectedEntities, simplemente coloque la columna adicional en un script personalizado.

API de Visual Job de AWS Glue

AWS Glue proporciona una API que permite a los clientes crear trabajos de integración de datos mediante la API de AWS Glue a partir de un objeto JSON que representa un flujo de trabajo de pasos visuales. Los clientes pueden utilizar el editor visual de AWS Glue Studio para realizar estos trabajos.

Para obtener más información sobre los tipos de datos de la API Visual Job, consulte API Visual Job.

Temas

- Diseño de API y las API de CRUD
- Introducción
- Limitaciones de los trabajos visuales

Diseño de API y las API de CRUD

Las API CreateJob y UpdateJob ahora son compatibles con un parámetro opcional adicional: codeGenConfigurationNodes. Proporcionar una estructura JSON no vacía para este campo hará que se registre el DAG en AWS Glue Studio para el trabajo creado y que se genere el código asociado. Se ignorará un valor nulo o una cadena vacía para este campo en la creación del trabajo.

Las actualizaciones del campo codeGenConfigurationNodes se realizarán a través de la API de AWS Glue UpdateJob de forma similar a CreateJob. En UpdateJob se debe especificar el campo completo en el que se ha modificado el DAG según lo deseado. Se ignorará todo valor nulo proporcionado y no se realizará ninguna actualización del DAG. Una estructura o cadena vacía hará que los codeGenConfigurationNodes se establezcan como vacíos y se elimine cualquier DAG anterior.

La API GetJob devolverá un DAG, si lo hay. La API DeleteJob también eliminará cualquier DAG asociado.

Introducción

Para crear un trabajo, utilice la acción CreateJob. La entrada de la solicitud `CreateJob` tendrá un campo “codeGenConfigurationNodes” adicional en el que se puede especificar el objeto DAG en JSON.

Para tener en cuenta:

- El campo “codeGenConfigurationNodes” es un mapa de nodeId a nodo.
- Cada nodo comienza con una clave que identifica qué tipo de nodo es.
- Solo puede haber una clave especificada, ya que un nodo solo puede ser de un tipo.
- El campo de entrada contiene los nodos principales del nodo actual.

La siguiente es una representación JSON de una entrada de CreateJob.

```
{
  "node-1": {
    "S3CatalogSource": {
      "Table": "csvFormattedTable",
      "PartitionPredicate": "",
      "Name": "S3 bucket",
      "AdditionalOptions": {},
      "Database": "myDatabase"
    }
  },
  "node-3": {
    "S3DirectTarget": {
      "Inputs": ["node-2"],
      "PartitionKeys": [],
      "Compression": "none",
      "Format": "json",
      "SchemaChangePolicy": { "EnableUpdateCatalog": false },
      "Path": "",
      "Name": "S3 bucket"
    }
  },
  "node-2": {
    "ApplyMapping": {
```

```
      "Inputs": ["node-1"],
      "Name": "ApplyMapping",
      "Mapping": [
        {
          "FromType": "long",
          "ToType": "long",
          "Dropped": false,
          "ToKey": "myheader1",
          "FromPath": ["myheader1"]
        },
        {
          "FromType": "long",
          "ToType": "long",
          "Dropped": false,
          "ToKey": "myheader2",
          "FromPath": ["myheader2"]
        },
        {
          "FromType": "long",
          "ToType": "long",
          "Dropped": false,
          "ToKey": "myheader3",
          "FromPath": ["myheader3"]
        }
      ]
    }
  }
}
```

Actualización y obtención de trabajos

Dado que UpdateJob también tendrá un campo “codeGenConfigurationNodes”, el formato de entrada será el mismo. Consulte Acción UpdateJob.

La acción GetJob devolverá también un campo “codeGenConfigurationNodes” con el mismo formato. Consulte Acción GetJob.

Limitaciones de los trabajos visuales

Dado que el parámetro “codeGenConfigurationNodes” se ha agregado a las API existentes, se heredará cualquier limitación en esas API. Además, los codeGenConfigurationNodes y

algunos nodos tendrán un tamaño limitado. Consulte Estructura de los trabajos para obtener más información.

Programación de scripts de Ray

AWS Glue facilita la escritura y ejecución de scripts de Ray. En esta sección se describen las capacidades de Ray compatibles que están disponibles en AWS Glue para Ray. Programación de scripts de Ray en Python.

El script personalizado debe ser compatible con la versión de Ray definida en el campo `Runtime` de la definición de su trabajo. Para obtener más información sobre `Runtime` en la API de los trabajos, consulte the section called “Trabajos”. Para obtener información acerca de cada entorno de tiempo de ejecución, consulte the section called “Entornos de tiempo de ejecución de Ray compatibles”.

Temas

- Tutorial: Cómo escribir un script ETL en AWS Glue para Ray
- Uso de Ray Core y Ray Data en AWS Glue para Ray
- Proporcionar archivos y bibliotecas de Python a los trabajos de Ray
- Conexión a los datos de los trabajos de Ray

Tutorial: Cómo escribir un script ETL en AWS Glue para Ray

Ray permite escribir y escalar tareas distribuidas de forma nativa en Python. AWS Glue para Ray ofrece entornos de Ray sin servidor a los que puede acceder tanto desde ambos trabajos y sesiones interactivas (las sesiones interactivas de Ray están en versión preliminar). El sistema de trabajo de AWS Glue proporciona una forma coherente de administrar y ejecutar las tareas de forma programada, mediante un disparador o desde la consola de AWS Glue.

La combinación de estas herramientas de AWS Glue crea una poderosa cadena de herramientas que puede usar para la extracción, transformación y carga (ETL) de cargas de trabajo, un caso de uso popular para AWS Glue. En este tutorial, aprenderá los conceptos básicos de la creación de esta solución.

También admitimos el uso de AWS Glue para Spark para sus cargas de trabajo de ETL. Para obtener acceso a un tutorial acerca de la escritura de un script de AWS Glue para Spark, consulte the section called “Tutorial: Escritura de un script de Spark”. Para obtener más información sobre

los motores disponibles, consulte the section called “AWS Glue para Spark y AWS Glue para Ray”. Ray es capaz de abordar muchos tipos diferentes de tareas en los ámbitos de la analítica, machine learning y el desarrollo de aplicaciones.

En este tutorial, extraerá, transformará y cargará un conjunto de datos de CSV alojado en Amazon Simple Storage Service (Amazon S3). Empezará con el conjunto de datos de registros de viajes de la Comisión de Taxis y Limusinas de Nueva York (TLC), que se almacena en un bucket público de Amazon S3. Para obtener más información sobre este conjunto de datos, consulte el Registro de datos abiertos en AWS.

Transformará sus datos con transformaciones predefinidas disponibles en la biblioteca Ray Data. Ray Data es una biblioteca de preparación de conjuntos de datos diseñada por Ray e incluida de forma predeterminada en los entornos de AWS Glue para Ray. Para obtener más información sobre las bibliotecas incluidad de forma predeterminada, consulte the section called “Módulos incluidos con los trabajos de Ray”. A continuación, escribirá los datos transformados en un bucket de Amazon S3 que usted controle.

Requisitos previos: para este tutorial, necesita una cuenta de AWS con acceso a AWS Glue y a Amazon S3.

Paso 1: crear un bucket en Amazon S3 para almacenar los datos de salida

Necesitará un bucket de Amazon S3 que pueda controlar para que sirva de depósito de los datos creados en este tutorial. Puede crear este bucket mediante el siguiente procedimiento.

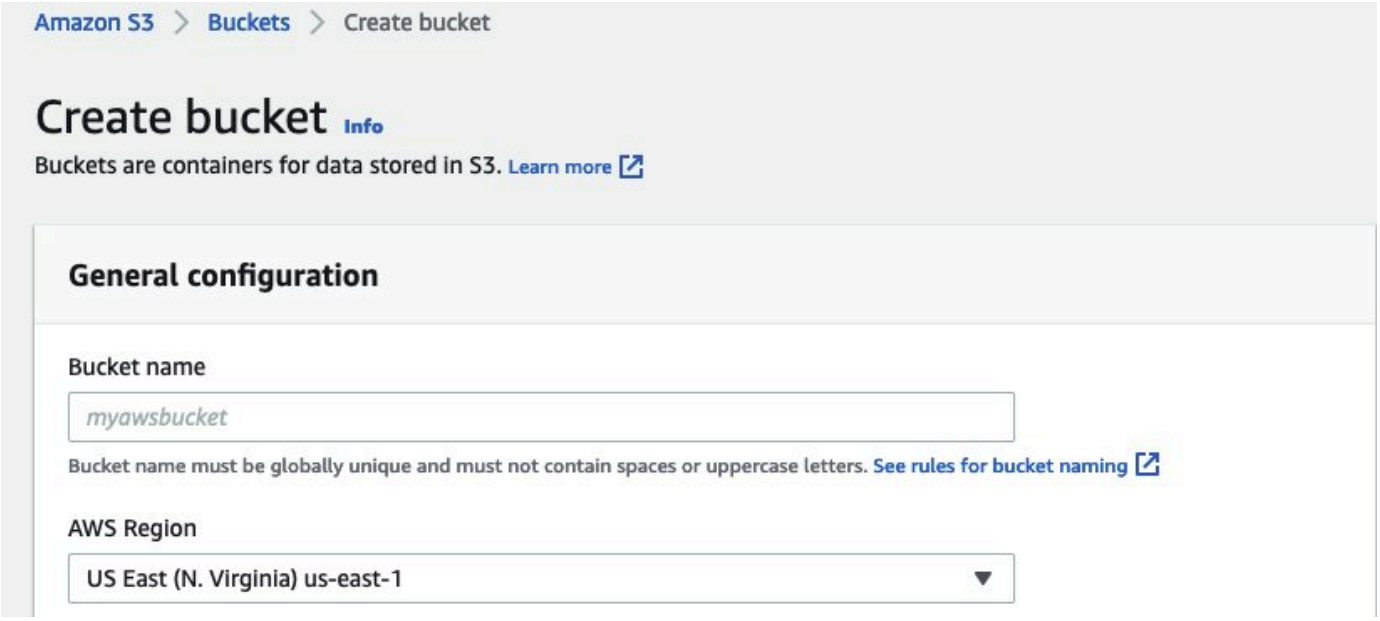
> **Note**
>
> Si quiere escribir sus datos en un bucket existente que usted controla, puede omitir este paso. Tome nota de *`yourBucketName`*, el nombre del bucket existente, para usarlo en pasos posteriores.

Para crear un bucket para la salida del trabajo de Ray

- Para crear un bucket, siga los pasos que se indican en Crear un bucket en la Guía del usuario de Amazon S3.
  - Cuando elija el nombre de un bucket, tome nota de *`yourBucketName`*, que consultará en pasos posteriores.

- Para otras configuraciones, la configuración sugerida que se proporciona en la consola de Amazon S3 debería funcionar correctamente en este tutorial.

Por ejemplo, el cuadro de diálogo de creación del bucket podría tener este aspecto en la consola de Amazon S3.



Paso 2: crear una política y rol de IAM para su trabajo de Ray

Su trabajo necesitará un rol de AWS Identity and Access Management (IAM) con lo siguiente:

- Permisos otorgados por la política administrada de `AWSGlueServiceRole`. Estos son los permisos básicos necesarios para ejecutar un trabajo de AWS Glue.
- `Read` los permisos de nivel de acceso para el recurso de Amazon S3 `nyc-tlc/*`.
- `Write` los permisos de nivel de acceso para el recurso de Amazon S3 *`yourBucketName`*`/*`.
- Una relación de confianza que permite a la entidad principal `glue.amazonaws.com` asumir el rol.

Puede crear este rol mediante el siguiente procedimiento.

Crear un rol de IAM para su trabajo de AWS Glue para Ray

> **Note**
>
> Puede seguir muchos procedimientos diferentes para crear un rol de IAM. Para obtener más información u opciones sobre cómo aprovisionar los recursos de IAM, consulte la documentación de AWS Identity and Access Management.

1. Cree una política que defina los permisos de Amazon S3 descritos anteriormente mediante los pasos que aparecen en Creación de políticas de IAM (consola) con el editor visual en la Guía del usuario de IAM.
   - Cuando seleccione un servicio, elija Amazon S3.
   - Cuando seleccione los permisos para su política, adjunte los siguientes conjuntos de acciones para los siguientes recursos (mencionados anteriormente):
     - Lea los permisos de nivel de acceso para el recurso de Amazon S3 `nyc-tlc/*`.
     - Escriba los permisos de nivel de acceso para el recurso de Amazon S3 *`yourBucketName`*`/*`.
   - Cuando seleccione el nombre de la política, tome nota de *`YourPolicyName`*, que consultará en un paso posterior.
2. Cree un rol para su trabajo de AWS Glue para Ray; para ello, siga los pasos que se indican en Crear un rol para un servicio de AWS (consola) en la Guía del usuario de IAM.
   - Cuando seleccione una entidad de servicio AWS de confianza, elija `Glue`. Esto completará automáticamente la relación de confianza necesaria para su trabajo.
   - Cuando seleccione políticas para la política de permisos, adjunte las siguientes políticas:
     - `AWSGlueServiceRole`
     - *`YourPolicyName`*
   - Cuando seleccione el nombre del rol, tome nota de *`YourRoleName`*, que consultará en pasos posteriores.

Paso 3: crear y ejecutar un trabajo de AWS Glue para Ray

En este paso, usted crea un trabajo de AWS Glue mediante la AWS Management Console, proporciona un script de ejemplo y ejecuta el trabajo. Cuando crea un trabajo, se crea un lugar en la

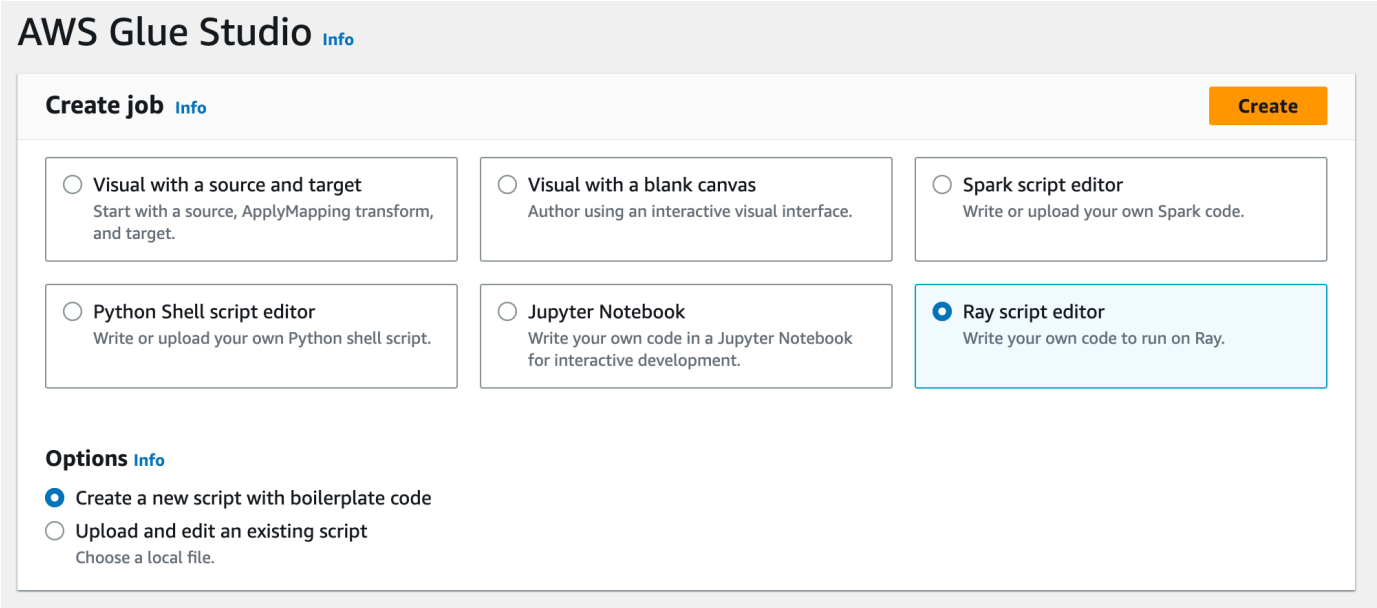
consola para que usted almacene, configure y edite el script de Ray. Para obtener más información acerca de la creación de trabajos, consulte the section called “Inicie sesión en la consola”.

En este tutorial, abordamos el siguiente escenario de ETL: lea los registros de enero de 2022 del conjunto de datos de Registros de viajes de TLC de New York City, agregue una nueva columna (`tip_rate`) al conjunto de datos mediante la combinación de datos en las columnas existentes; luego, elimine la cantidad de columnas que no sean relevantes para el análisis actual y, a continuación, escriba los resultados en *`yourBucketName`*. El siguiente script de Ray lleva a cabo estos pasos:

```
import ray
import pandas
from ray import data

ray.init('auto')

ds = ray.data.read_csv("s3://nyc-tlc/opendata_repo/opendata_webconvert/yellow/
yellow_tripdata_2022-01.csv")

# Add the given new column to the dataset and show the sample record after adding a new
 column
ds = ds.add_column( "tip_rate", lambda df: df["tip_amount"] / df["total_amount"])

# Dropping few columns from the underlying Dataset
ds = ds.drop_columns(["payment_type", "fare_amount", "extra", "tolls_amount",
 "improvement_surcharge"])

ds.write_parquet("s3://yourBucketName/ray/tutorial/output/")
```

Para crear y ejecutar un trabajo de AWS Glue para Ray

1. En la AWS Management Console, vaya a la página de inicio de AWS Glue.
2. En el panel de navegación lateral, seleccione Trabajos de ETL.
3. En Crear trabajo, elija Editor de script de Ray y, a continuación, elija Crear, como se muestra en la siguiente ilustración.



4. Pegue el texto completo del script en el panel Script y sustituya el texto existente.
5. Vaya a Detalles del trabajo y configure la propiedad de rol de IAM en *YourRoleName*.
6. Elija Guardar y, a continuación, elija Ejecutar.

Paso 4: inspeccionar la salida

Después de ejecutar su trabajo de AWS Glue, debe validar que el resultado cumpla con las expectativas de este escenario. Puede hacerlo mediante el siguiente procedimiento.

Para validar si su trabajo de Ray se ejecutó correctamente

1. En la página de detalles del trabajo, vaya a Ejecuciones.
2. Después de unos minutos, debería ver una ejecución con el Estado de ejecución como Correcto.
3. Vaya a la consola de Amazon S3 en https://console.aws.amazon.com/s3/ e inspeccione *yourBucketName*. Debería ver los archivos escritos en su bucket de salida.
4. Lea los archivos de Parquet y verifique su contenido. Puede hacerlo con las herramientas existentes. Si no dispone de un proceso para validar los archivos de Parquet, puede hacerlo en la consola de AWS Glue mediante una sesión interactiva de AWS Glue, con Spark o Ray (versión preliminar).

   En una sesión interactiva, tiene acceso a las bibliotecas Ray Data, Spark o pandas, que se proporcionan de forma predeterminada (según el motor que elija). Para verificar el contenido del archivo, puede usar los métodos de inspección habituales disponibles en esas bibliotecas,

como `count`, `schema` y `show`. Para obtener más información sobre las sesiones interactivas en la consola, consulte Uso de cuaderno con AWS Glue Studio y AWS Glue.

Como ha confirmado que los archivos se han escrito en el bucket, puede decir con relativa certeza que si el resultado tiene problemas, no están relacionados con la configuración de IAM. Configure su sesión con *`YourRoleName`* para tener acceso a los archivos pertinentes.

Si no ve los resultados esperados, examine el contenido de resolución de problemas que aparecen en esta guía para identificar y corregir el origen del error. Para interpretar los estados de error al ejecutar un trabajo, consulte the section called “Estados de ejecuciones de trabajos”. Encontrará el contenido sobre la resolución de problemas en el capítulo Solución de problemas de AWS Glue. Para ver errores específicos relacionados con los trabajos de Ray, consulte the section called “Solución de errores de Ray” en el capítulo sobre resolución de problemas.

Siguientes pasos

Ya ha visto y realizado un proceso de ETL mediante AWS Glue para Ray de principio a fin. Puede utilizar los siguientes recursos a fin de comprender qué herramientas proporciona AWS Glue para Ray para transformar e interpretar sus datos a escala.

- Para obtener más información sobre este tema, consulte the section called “Uso de Ray Core y Ray Data en AWS Glue para Ray”. Para obtener más experiencia en el uso de las tareas de Ray, siga los ejemplos que aparecen en la documentación de Ray Core. Consulte Ray Core: Ray Tutorials and Examples (2.4.0) en la documentación de Ray.
- Para obtener información sobre las bibliotecas de administración de datos disponibles en AWS Glue para Ray, consulte the section called “Conexión a datos”. Para obtener más experiencia en el uso de Ray Data para transformar y escribir conjuntos de datos, siga los ejemplos que aparecen en la documentación de Ray Data. Consulte Ray Data: Examples (2.4.0).
- Para obtener más información sobre la configuración de trabajos de AWS Glue para Ray, consulte the section called “Trabajar con tareas de Ray”.
- Para obtener más información sobre cómo escribir scripts de AWS Glue para Ray, continúe leyendo la documentación de esta sección.

Uso de Ray Core y Ray Data en AWS Glue para Ray

Ray es un marco para escalar verticalmente los scripts de Python mediante la distribución del trabajo en un clúster. Puede usar Ray como solución a muchos tipos de problemas, por lo que Ray

proporciona bibliotecas para optimizar determinadas tareas. En AWS Glue, nos centramos en usar Ray para transformar grandes conjuntos de datos. AWS Glueofrece soporte para Ray Data y partes de Ray Core a fin de facilitar esta tarea.

¿Qué es Ray Core?

El primer paso para crear una aplicación distribuida es identificar y definir el trabajo que se puede realizar de forma simultánea. Ray Core contiene las partes de Ray que se utilizan para definir las tareas que se pueden realizar de forma simultánea. Ray proporciona información de referencia y de inicio rápido que puede utilizar para aprender las herramientas que proporcionan. Para obtener más información, consulte ¿Qué es Ray Core? y Ray Core Quick Start. Para obtener más información sobre cómo definir eficazmente las tareas simultáneas en Ray, consulte Consejos para usuarios principiantes.

> **Tareas y actores de Ray**
>
> En la documentación de AWS Glue para Ray, podríamos referirnos a las tareas y los actores, que son conceptos básicos de Ray.
> Ray usa las características y clases de Python como bloques de construcción de un sistema de computación distribuida. Al igual que cuando las características y variables de Python se convierten en “métodos” y “atributos” cuando se usan en una clase, las funciones se convierten en “tareas” y las clases se convierten en “actores” cuando se usan en Ray para enviar código a los trabajadores. Mediante la anotación de `@ray.remote`, puede identificar las características y las clases que Ray podría usar.
> Las tareas y los actores son configurables, tienen un ciclo de vida y consumen recursos informáticos a lo largo de su vida. El código que genera errores se puede rastrear hasta una tarea o actor cuando se busca la causa raíz de los problemas. Por lo tanto, estos términos pueden surgir cuando aprenda a configurar, monitorear o depurar trabajos de AWS Glue para Ray.
> Para empezar a aprender a utilizar eficazmente las tareas y los actores a fin de crear una aplicación distribuida, consulte Conceptos clave en los documentos de Ray.

Ray Core en AWS Glue para Ray

AWS Glue para los entornos de Ray administra la formación y el escalado vertical de los clústeres, así como la recopilación y visualización de registros. Como gestionamos estos problemas, limitamos

el acceso y el soporte a las API de Ray Core que se utilizarían para abordar estos problemas en un clúster de código abierto.

En el entorno de tiempo de ejecución de `Ray2.4` administrado, no admitimos lo siguiente:

- CLI de Ray Core
- CLI de Ray State
- `ray.util.metrics`Métodos de utilidad métrica de Prometheus:
  - Contador
  - Calibre
  - Histograma
- Otras herramientas de depuración:
  - ray.util.pdb.set_trace
  - ray.util.inspect_serializabilidad
  - ray.timeline

¿Qué es Ray Data?

Cuando se conecta a orígenes y destinos de datos, gestiona conjuntos de datos e inicia transformaciones comunes, Ray Data es una metodología sencilla para usar Ray a fin de resolver problemas mediante la transformación de conjuntos de datos de Ray. Para obtener más información sobre el uso de Ray Data, consulte Ray Datasets: Preprocesamiento de datos distribuidos.

Puede usar Ray Data u otras herramientas para acceder a sus datos. Para obtener más información sobre cómo acceder a sus datos en Ray, consulte the section called “Conexión a datos”.

Ray Data en AWS Glue para Ray

Ray Data es compatible y se proporciona de forma predeterminada en el entorno de tiempo de ejecución de `Ray2.4` administrado. Para obtener más información sobre los módulos proporcionados, consulte the section called “Módulos incluidos con los trabajos de Ray”.

Proporcionar archivos y bibliotecas de Python a los trabajos de Ray

En esta sección, se proporciona información que se necesita para utilizar las bibliotecas de Python con los trabajos de Ray de AWS Glue. Puede utilizar determinadas bibliotecas comunes incluidas

de forma predeterminada en todos los trabajos de Ray. También puede proporcionar sus propias bibliotecas de Python en su trabajo de Ray.

Módulos incluidos con los trabajos de Ray

Puede aplicar flujos de trabajo de integración de datos en un trabajo de Ray con los siguientes paquetes proporcionados. Estos paquetes están disponibles de manera predeterminada en trabajos de Ray.

AWS Glue version 4.0

En AWS Glue 4.0, el entorno de Ray (tiempo de ejecución `Ray2.4`) proporciona los siguientes paquetes:

- boto3 == 1.26.133
- ray == 2.4.0
- pyarrow == 11.0.0
- pandas == 1.5.3
- numpy == 1.24.3
- fsspec == 2023.4.0

Esta lista incluye todos los paquetes que se instalarían con `ray[data] == 2.4.0`. Ray Data se admite de fábrica.

Proporcionar archivos a su trabajo de Ray

Puede proporcionar archivos a su trabajo de Ray con el parámetro `--working-dir`. Proporcione este parámetro con una ruta a un archivo .zip alojado en Amazon S3. Dentro del archivo .zip, los archivos deben estar contenidos en un único directorio de nivel superior. No debe haber ningún otro archivo en el nivel superior.

Los archivos se distribuirán a cada nodo de Ray antes de que el script comience a ejecutarse. Considere cómo esto podría repercutir en el espacio en disco disponible para cada nodo de Ray. El espacio en disco disponible viene determinado por el WorkerType establecido en la configuración del trabajo. Si desea proporcionar los datos de su trabajo a escala, este mecanismo no es la solución adecuada. Para obtener más información sobre cómo proporcionar datos a sus trabajos, consulte the section called “Conexión a datos”.

Se podrá acceder a sus archivos como si el directorio se hubiera proporcionado a Ray a través del parámetro `working_dir`. Por ejemplo, para leer un archivo denominado `sample.txt` en el directorio de nivel superior del archivo .zip, puede llamar a:

```
@ray.remote
def do_work():
    f = open("sample.txt", "r")
    print(f.read())
```

Para obtener más información acerca de `working_dir`, consulte la documentación de Ray. Esta característica se comporta de forma similar a las capacidades nativas de Ray.

Módulos de Python adicionales para trabajos de Ray

Módulos adicionales de PyPI

Los trabajos de Ray utilizan el instalador de paquetes de Python (pip3) para instalar los módulos adicionales que utilizará un script de Ray. Puede utilizar el parámetro `--pip-install` con una lista de módulos de Python separados por comas para agregar un nuevo módulo o cambiar la versión de un módulo existente.

Por ejemplo, para actualizar o agregar un nuevo módulo `scikit-learn`, utilice el siguiente par clave-valor:

`"--pip-install", "scikit-learn==0.21.3"`

Si tiene módulos o parches personalizados, puede distribuir sus propias bibliotecas desde Amazon S3 con el parámetro `--s3-py-modules`. Es posible que sea necesario volver a empaquetar y compilar su distribución antes de cargarla. Seguir las directrices en the section called “Incluir código Python en los trabajos de Ray”.

Distribuciones personalizadas de Amazon S3

Las distribuciones personalizadas deben cumplir con las pautas de empaquetado de Ray para las dependencias. Puede encontrar información sobre la compilación de estas distribuciones en la siguiente sección. Para obtener más información sobre cómo Ray configura las dependencias, consulte Environment Dependencies (Dependencias de entorno) en la documentación de Ray.

Para incluir un distribuible personalizado después de evaluar su contenido, cárguelo en un bucket disponible para el rol de IAM del trabajo. Especifique la ruta de Amazon S3 a un archivo zip de

Python en la configuración del parámetro. Si proporciona varios distribuibles, sepárelos con comas. Por ejemplo:

```
"--s3-py-modules", "s3://s3bucket/pythonPackage.zip"
```

Limitaciones

Los trabajos de Ray no admiten la compilación de código nativo en el entorno de trabajo. Esto puede ser una limitación si las dependencias de Python dependen de forma transitiva de código nativo compilado. Los trabajos de Ray pueden ejecutar los binarios proporcionados, pero deben compilarse para Linux en ARM64. Esto significa que quizás pueda utilizar el contenido de las ruedas de `aarch64manylinux`. Para poder proporcionar las dependencias nativas en un formato compilado, vuelva a empaquetar una rueda según los estándares de Ray. Por lo general, esto significa eliminar las carpetas `dist-info` para que solo haya una carpeta en la raíz del archivo.

No puede actualizar la versión de `ray` o `ray[data]` mediante este parámetro. Para usar una nueva versión de Ray, tendrá que cambiar el campo de tiempo de ejecución de su trabajo una vez que hayamos publicado el soporte para ello. Para obtener más información acerca de las versiones de Ray admitidas, consulte the section called “Versiones de AWS Glue”.

Incluir código Python en los trabajos de Ray

Python Software Foundation ofrece comportamientos estandarizados para empaquetar archivos de Python para su uso en diferentes entornos de ejecución. Ray introduce limitaciones en los estándares de empaquetado que debe conocer. AWS Glue no especifica estándares de empaquetado más allá de los especificados para Ray. Las siguientes instrucciones proporcionan una guía estándar sobre cómo empaquetar paquetes simples de Python.

Empaquete sus archivos en un archivo `.zip`. El directorio debe estar en la raíz del archivo. No debe haber ningún otro archivo en el nivel raíz del archivo; de lo contrario, se producirá un comportamiento inesperado. El directorio raíz es el paquete, y su nombre se utiliza para hacer referencia al código de Python al importarlo.

Si proporciona una distribución de esta forma a un trabajo de Ray con `--s3-py-modules`, podrá importar código de Python del paquete en el script de Ray.

Su paquete puede proporcionar un único módulo de Python con algunos archivos de Python o puede empaquetar varios módulos juntos. Al volver a empaquetar las dependencias, como las bibliotecas de PyPI, compruebe si hay archivos ocultos y directorios de metadatos dentro de esos paquetes.

> **Warning**
>
> Determinados comportamientos del sistema operativo dificultan el seguimiento correcto de estas instrucciones de empaquetado.
>
> - OSX puede agregar archivos ocultos, como `__MACOSX`, a su archivo zip en el nivel superior.
> - Windows puede agregar sus archivos a una carpeta dentro del zip automáticamente y crear involuntariamente una carpeta anidada.

En los siguientes procedimientos se supone que interactúa con los archivos en Amazon Linux 2 o en un sistema operativo similar que proporciona una distribución de las utilidades `zip` y `zipinfo` de Info-ZIP. Recomendamos utilizar estas herramientas para evitar comportamientos inesperados.

Para empaquetar archivos de Python para usarlos en Ray

1. Cree un directorio temporal con el nombre de su paquete; a continuación, confirme que su directorio de trabajo es el directorio principal. Puede hacer esto con los siguientes comandos:

   ```
   cd parent_directory
   mkdir temp_dir
   ```

2. Copie los archivos en el directorio temporal; a continuación, confirme la estructura del directorio. Se accederá directamente al contenido de este directorio como módulo de Python. Puede hacer esto con el siguiente comando de la :

   ```
   ls -AR temp_dir
   # my_file_1.py
   # my_file_2.py
   ```

3. Use zip para comprimir la carpeta temporal. Puede hacer esto con los siguientes comandos:

   ```
   zip -r zip_file.zip temp_dir
   ```

4. Confirme que el archivo esté correctamente empaquetado. Ahora, *`zip_file`*`.zip` debería encontrarse en su directorio de trabajo. Puede inspeccionarlo con el siguiente comando:

   ```
   zipinfo -1 zip_file.zip
   ```

```
# temp_dir/
# temp_dir/my_file_1.py
# temp_dir/my_file_2.py
```

Volver a empaquetar un paquete de Python para usarlo en Ray.

1. Cree un directorio temporal con el nombre de su paquete; a continuación, confirme que su directorio de trabajo es el directorio principal. Puede hacer esto con los siguientes comandos:

```
cd parent_directory
mkdir temp_dir
```

2. Descomprima el paquete y copie el contenido en el directorio temporal. Elimine los archivos relacionados con su estándar de empaquetado anterior y deje solo el contenido del módulo. Confirme que la estructura de archivos sea correcta con el siguiente comando:

```
ls -AR temp_dir
# my_module
# my_module/__init__.py
# my_module/my_file_1.py
# my_module/my_submodule/__init__.py
# my_module/my_submodule/my_file_2.py
# my_module/my_submodule/my_file_3.py
```

3. Use zip para comprimir la carpeta temporal. Puede hacer esto con los siguientes comandos:

```
zip -r zip_file.zip temp_dir
```

4. Confirme que el archivo esté correctamente empaquetado. Ahora, *zip_file*`.zip` debería encontrarse en su directorio de trabajo. Puede inspeccionarlo con el siguiente comando:

```
zipinfo -1 zip_file.zip
# temp_dir/my_module/
# temp_dir/my_module/__init__.py
# temp_dir/my_module/my_file_1.py
# temp_dir/my_module/my_submodule/
# temp_dir/my_module/my_submodule/__init__.py
# temp_dir/my_module/my_submodule/my_file_2.py
# temp_dir/my_module/my_submodule/my_file_3.py
```

Conexión a los datos de los trabajos de Ray

Los trabajos de Ray de AWS Glue pueden usar una amplia matriz de paquetes de Python que están diseñados para que usted pueda integrar datos rápidamente. Proporcionamos un conjunto mínimo de dependencias para no saturar su entorno. Para obtener más información acerca de lo que se incluye de forma predeterminada, consulte the section called “Módulos incluidos con los trabajos de Ray”.

> **Note**
>
> Extracción, transformación y carga (ETL) de AWS Glue proporciona la abstracción de DynamicFrame para simplificar los flujos de trabajo de ETL en los que se resuelven las diferencias de esquema entre las filas en el conjunto de datos. AWS Glue ETL ofrece características adicionales: marcadores de trabajos y agrupación de archivos entrada. Por el momento, no proporcionamos las características correspondientes en los trabajos de Ray. AWS Glue para Spark ofrece asistencia directa para conectarse a determinados formatos, orígenes y receptores de datos. En Ray, AWS SDK para pandas y bibliotecas externas actuales cubren sustancialmente esa necesidad. Deberá consultar esas bibliotecas para conocer las capacidades disponibles.

La integración de AWS Glue para Ray con Amazon VPC no está disponible actualmente. No se podrá acceder a los recursos de Amazon VPC sin una ruta pública. Para obtener más información acerca del uso de AWS Glue con Amazon VPC, consulte the section called “Puntos de conexión de VPC (AWS PrivateLink)”.

Bibliotecas comunes para trabajar con datos en Ray

Ray Data: Ray Data proporciona métodos para gestionar formatos, orígenes y receptores de datos comunes. Para obtener más información sobre los formatos y orígenes compatibles con Ray Data, consulte Input/Output en la documentación de Ray Data. Ray Data es una biblioteca obstinada, en lugar de una biblioteca de uso general, para gestionar conjuntos de datos.

Ray proporciona cierta orientación sobre los casos de uso en los que Ray Data podría ser la mejor solución para su trabajo. Para obtener más información, consulte Casos de uso de Ray en la documentación de Ray.

AWS SDK para pandas (awswrangler): AWS SDK es un producto de AWS que ofrece soluciones limpias y probadas para leer y escribir en los servicios de AWS cuando las transformaciones

administran datos con DataFrames de pandas. Para obtener más información sobre los formatos y orígenes compatibles con AWS SDK para pandas, consulte API Reference en la documentación de AWS SDK para pandas.

Para ver ejemplos de cómo leer y escribir datos con AWS SDK para pandas, consulte Quick Start en la documentación de AWS SDK para pandas. AWS SDK para pandas no proporciona transformaciones para sus datos. Solo proporciona asistencia para leer y escribir desde las orígenes.

Modin: Modin es una biblioteca de Python que implementa operaciones comunes de pandas de forma distribuible. Para obtener más información sobre Modin, consulte la documentación de Mondin. Modin en sí no admite la lectura y la escritura a partir de orígenes. Proporciona implementaciones distribuidas de transformaciones comunes. AWS SDK para pandas admite Modin.

Cuando ejecuta Modin y AWS SDK para pandas juntos en un entorno de Ray, puede realizar tareas ETL habituales con resultados de alto rendimiento. Para obtener más información sobre el uso de Modin con AWS SDK para pandas, consulte At scale en la documentación de AWS SDK para pandas.

Otros esquemas: para obtener más información sobre los esquemas compatibles con Ray, consulte El ecosistema de Ray en la documentación de Ray. No ofrecemos soporte para otros marcos en el caso AWS Glue para Ray.

Conexión a los datos mediante el Catálogo de datos

AWS SDK para pandas admite la administración de los datos a través del Catálogo de datos junto con los trabajos de Ray. Para más información, consulte Glue Catalog (Catálogo de Glue) en el sitio web de AWS SDK para pandas.

Uso de este servicio con un AWS SDK

AWS Los kits de desarrollo de software (SDK) están disponibles para muchos lenguajes de programación populares. Cada SDK proporciona una API, ejemplos de código y documentación que facilitan a los desarrolladores la creación de aplicaciones en su lenguaje preferido.

Documentación de SDK	Ejemplos de código
AWS SDK for C++	AWS SDK for C++ ejemplos de código
AWS CLI	AWS CLI ejemplos de código
AWS SDK for Go	AWS SDK for Go ejemplos de código
AWS SDK for Java	AWS SDK for Java ejemplos de código
AWS SDK for JavaScript	AWS SDK for JavaScript ejemplos de código
AWS SDK para Kotlin	AWS SDK para Kotlin ejemplos de código
AWS SDK for .NET	AWS SDK for .NET ejemplos de código
AWS SDK for PHP	AWS SDK for PHP ejemplos de código
AWS Tools for PowerShell	Herramientas para ejemplos PowerShell de código
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) ejemplos de código
AWS SDK for Ruby	AWS SDK for Ruby ejemplos de código
AWS SDK para Rust	AWS SDK para Rust ejemplos de código
AWS SDK para SAP ABAP	AWS SDK para SAP ABAP ejemplos de código
AWS SDK para Swift	AWS SDK para Swift ejemplos de código

Para obtener ejemplos específicos de este servicio, consulte AWS Glue Ejemplos de código de API con AWS SDK.

> **Ejemplo de disponibilidad**
>
> ¿No encuentra lo que necesita? Solicite un ejemplo de código a través del enlace de Enviar comentarios que se encuentra al final de esta página.

AWS Glue API

En esta sección, se describen los tipos de datos y las primitivas que utilizan los AWS Glue SDK y las herramientas. Existen tres formas generales de interactuar AWS Glue programáticamente fuera del AWS Management Console, cada una con su propia documentación:

- Las bibliotecas de SDK de idiomas le permiten acceder a recursos de AWS de lenguajes de programación comunes. Encuentre más información en Herramientas para crear en AWS.
- AWS CLI Le permite acceder a los AWS recursos desde la línea de comandos. Encuentre más información en Referencia de comandos de la AWS CLI.
- AWS CloudFormation le permite definir un conjunto de AWS recursos que se aprovisionarán juntos de forma coherente. Encontrará más información en AWS CloudFormation: referencia del tipo de AWS Glue recurso.

En esta sección se registran las primitivas compartidas independientemente de estos SDK y herramientas. Las herramientas utilizan la referencia de la API AWS Glue web para comunicarse con AWS.

Contenido

- API de seguridad en AWS Glue
  - Tipos de datos
  - DataCatalogEncryptionSettings structure
  - Estructura EncryptionAtRest
  - Estructura ConnectionPasswordEncryption
  - Estructura EncryptionConfiguration
  - Estructura S3Encryption
  - Estructura CloudWatchEncryption
  - Estructura JobBookmarksEncryption
  - Estructura SecurityConfiguration
  - Estructura GluePolicy
  - Operaciones
  - Acción GetDataCatalogEncryptionSettings (Python: get_data_catalog_encryption_settings)
  - Acción PutDataCatalogEncryptionSettings (Python: put_data_catalog_encryption_settings)

- Acción PutResourcePolicy (Python: put_resource_policy)
- Acción GetResourcePolicy (Python: get_resource_policy)
- Acción DeleteResourcePolicy (Python: delete_resource_policy)
- Acción CreateSecurityConfiguration (Python: create_security_configuration)
- Acción DeleteSecurityConfiguration (Python: delete_security_configuration)
- Acción GetSecurityConfiguration (Python: get_security_configuration)
- Acción GetSecurityConfigurations (Python: get_security_configurations)
- Acción GetResourcePolicies (Python: get_resource_policies)
- API del catálogo
  - API de la base de datos
    - Tipos de datos
    - Estructura de base de datos
    - Estructura DatabaseInput
    - Estructura PrincipalPermissions
    - Estructura DataLakePrincipal
    - Estructura DatabaseIdentifier
    - Estructura de FederatedDatabase
    - Operaciones
    - Acción CreateDatabase (Python: create_database)
    - Acción UpdateDatabase (Python: update_database)
    - Acción DeleteDatabase (Python: delete_database)
    - Acción GetDatabase (Python: get_database)
    - Acción GetDatabases (Python: get_databases)
  - API de tabla
    - Tipos de datos
    - Estructura de tabla
    - TableInput estructura
    - FederatedTable estructura
    - Estructura de las columnas
    - StorageDescriptor estructura

- SchemaReference estructura
- SerDeInfo estructura
- Estructura Order
- SkewedInfo estructura
- TableVersion estructura
- TableError estructura
- TableVersionError estructura
- SortCriterion estructura
- TableIdentifier estructura
- KeySchemaElement estructura
- PartitionIndex estructura
- PartitionIndexDescriptor estructura
- BackfillError estructura
- IcebergInput estructura
- OpenTableFormatInput estructura
- ViewDefinition estructura
- ViewDefinitionInput estructura
- ViewRepresentation estructura
- ViewRepresentationInput estructura
- Operaciones
- CreateTable acción (Python: create_table)
- UpdateTable acción (Python: update_table)
- DeleteTable acción (Python: delete_table)
- BatchDeleteTable acción (Python: batch_delete_table)
- GetTable acción (Python: get_table)
- GetTables acción (Python: get_tables)
- GetTableVersion acción (Python: get_table_version)
- GetTableVersions acción (Python: get_table_versions)
- DeleteTableVersion acción (Python: delete_table_version)
- BatchDeleteTableVersion acción (Python: batch_delete_table_version)

- SearchTables acción (Python: search_tables)
- GetPartitionIndexes acción (Python: get_partition_indexes)
- CreatePartitionIndex acción (Python: create_partition_index)
- DeletePartitionIndex acción (Python: delete_partition_index)
- GetColumnStatisticsForTable acción (Python: get_column_statistics_for_table)
- UpdateColumnStatisticsForTable acción (Python: update_column_statistics_for_table)
- DeleteColumnStatisticsForTable acción (Python: delete_column_statistics_for_table)

- API de partición
  - Tipos de datos
  - Estructura Partition
  - Estructura PartitionInput
  - Estructura PartitionSpecWithSharedStorageDescriptor
  - Estructura PartitionListComposingSpec
  - Estructura PartitionSpecProxy
  - Estructura PartitionValueList
  - Estructura Segment
  - Estructura PartitionError
  - Estructura BatchUpdatePartitionFailureEntry
  - Estructura BatchUpdatePartitionRequestEntry
  - Estructura StorageDescriptor
  - Estructura SchemaReference
  - Estructura SerDeInfo
  - Estructura SkewedInfo
  - Operaciones
  - Acción CreatePartition (Python: create_partition)
  - Acción BatchCreatePartition (Python: batch_create_partition)
  - Acción UpdatePartition (Python: update_partition)
  - Acción DeletePartition (Python: delete_partition)
  - Acción BatchDeletePartition (Python: batch_delete_partition)
  - Acción GetPartition (Python: get_partition)

- Acción GetPartitions (Python: get_partitions)
- Acción BatchGetPartition (Python: batch_get_partition)
- Acción BatchUpdatePartition (Python: batch_update_partition)
- Acción GetColumnStatisticsForPartition (Python: get_column_statistics_for_partition)
- Acción UpdateColumnStatisticsForPartition (Python: update_column_statistics_for_partition)
- Acción DeleteColumnStatisticsForPartition (Python: delete_column_statistics_for_partition)

- API de conexión
  - Tipos de datos
  - Estructura Connection
  - Estructura ConnectionInput
  - Estructura PhysicalConnectionRequirements
  - Estructura GetConnectionsFilter
  - Operaciones
  - Acción CreateConnection (Python: create_connection)
  - Acción DeleteConnection (Python: delete_connection)
  - Acción GetConnection (Python: get_connection)
  - Acción GetConnections (Python: get_connections)
  - Acción UpdateConnection (Python: update_connection)
  - Acción BatchDeleteConnection (Python: batch_delete_connection)
  - Configuración de la autenticación
  - Estructura AuthenticationConfiguration
  - Estructura AuthenticationConfigurationInput
  - Estructura OAuth2Properties
  - Estructura OAuth2PropertiesInput
  - Estructura OAuth2ClientApplication
  - Estructura AuthorizationCodeProperties
- API de funciones definidas por el usuario
  - Tipos de datos
  - Estructura UserDefinedFunction
  - Estructura UserDefinedFunctionInput

    - Operaciones
    - Acción CreateUserDefinedFunction (Python: create_user_defined_function)
    - Acción UpdateUserDefinedFunction (Python: update_user_defined_function)
    - Acción DeleteUserDefinedFunction (Python: delete_user_defined_function)
    - Acción GetUserDefinedFunction (Python: get_user_defined_function)
    - Acción GetUserDefinedFunctions (Python: get_user_defined_functions)
  - Importación de un catálogo de Athena a AWS Glue
    - Tipos de datos
    - Estructura CatalogImportStatus
    - Operaciones
    - Acción ImportCatalogToGlue (Python: import_catalog_to_glue)
    - Acción GetCatalogImportStatus (Python: get_catalog_import_status)
- API del optimizador de tablas
  - Tipos de datos
  - TableOptimizer estructura
  - TableOptimizerConfiguration estructura
  - TableOptimizerRun estructura
  - RunMetrics estructura
  - BatchGetTableOptimizerEntry estructura
  - BatchTableOptimizer estructura
  - BatchGetTableOptimizerError estructura
  - Operaciones
  - GetTableOptimizer acción (Python: get_table_optimizer)
  - BatchGetTableOptimizer acción (Python: batch_get_table_optimizer)
  - ListTableOptimizerRuns acción (Python: list_table_optimizer_runs)
  - CreateTableOptimizer acción (Python: create_table_optimizer)
  - DeleteTableOptimizer acción (Python: delete_table_optimizer)
  - UpdateTableOptimizer acción (Python: update_table_optimizer)
- API de rastreadores y clasificadores
  - API de clasificador

- Tipos de datos
- Estructura Classifier
- Estructura GrokClassifier
- Estructura XMLClassifier
- Estructura JsonClassifier
- Estructura CsvClassifier
- Estructura CreateGrokClassifierRequest
- Estructura UpdateGrokClassifierRequest
- Estructura CreateXMLClassifierRequest
- Estructura UpdateXMLClassifierRequest
- Estructura CreateJsonClassifierRequest
- Estructura UpdateJsonClassifierRequest
- Estructura CreateCsvClassifierRequest
- Estructura UpdateCsvClassifierRequest
- Operaciones
- Acción CreateClassifier (Python: create_classifier)
- Acción DeleteClassifier (Python: delete_classifier)
- Acción GetClassifier (Python: get_classifier)
- Acción GetClassifiers (Python: get_classifiers)
- Acción UpdateClassifier (Python: update_classifier)

- La API del rastreador
  - Tipos de datos
  - Estructura de rastreador
  - Estructura de programación
  - CrawlerTargets estructura
  - Estructura S3Target
  - DeltaCatalogTarget Estructura S3
  - DeltaDirectTarget Estructura S3
  - JdbcTarget estructura
  - Estructura MongoDBTarget

- Estructura DynamoDBTarget
- DeltaTarget estructura
- IcebergTarget estructura
- HudiTarget estructura
- CatalogTarget estructura
- CrawlerMetrics estructura
- CrawlerHistory estructura
- CrawlsFilter estructura
- SchemaChangePolicy estructura
- LastCrawlInfo estructura
- RecrawlPolicy estructura
- LineageConfiguration estructura
- LakeFormationConfiguration estructura
- Operaciones
- CreateCrawler acción (Python: create_crawler)
- DeleteCrawler acción (Python: delete_crawler)
- GetCrawler acción (Python: get_crawler)
- GetCrawlers acción (Python: get_crawlers)
- GetCrawlerMetrics acción (Python: get_crawler_metrics)
- UpdateCrawler acción (Python: update_crawler)
- StartCrawler acción (Python: start_crawler)
- StopCrawler acción (Python: stop_crawler)
- BatchGetCrawlers acción (Python: batch_get_crawlers)
- ListCrawlers acción (Python: list_crawlers)
- ListCrawls acción (Python: list_crawls)
- API de estadísticas de columna
  - Tipos de datos
  - Estructura ColumnStatisticsTaskRun
  - Estructura ColumnStatisticsTaskRunningException
  - Estructura ColumnStatisticsTaskNotRunningException

- Estructura ColumnStatisticsTaskStopingException
- Operaciones
- StartColumnStatisticsTaskRun action (Python: start_column_statistics_task_run)
- GetColumnStatisticsTaskRun action (Python: get_column_statistics_task_run)
- GetColumnStatisticsTaskRuns action (Python: get_column_statistics_task_runs)
- ListColumnStatisticsTaskRuns action (Python: list_column_statistics_task_runs)
- StopColumnStatisticsTaskRun action (Python: stop_column_statistics_task_run)

- API del programador del rastreador
  - Tipos de datos
  - Estructura de programación
  - Operaciones
  - Acción UpdateCrawlerSchedule (Python: update_crawler_schedule)
  - Acción StartCrawlerSchedule (Python: start_crawler_schedule)
  - Acción StopCrawlerSchedule (Python: stop_crawler_schedule)

- API de generación automática de scripts de ETL
  - Tipos de datos
  - Estructura CodeGenNode
  - Estructura CodeGenNodeArg
  - Estructura CodeGenEdge
  - Estructura de ubicación
  - Estructura CatalogEntry
  - Estructura MappingEntry
  - Operaciones
  - Acción CreateScript (Python: create_script)
  - Acción GetDataflowGraph (Python: get_dataflow_graph)
  - Acción GetMapping (Python: get_mapping)
  - Acción GetPlan (Python: get_plan)

- API de Visual Job
  - Tipos de datos
  - CodeGenConfigurationNode estructura

- Estructura JDBC ConnectorOptions
- StreamingDataPreviewOptions estructura
- AthenaConnectorSource estructura
- Estructura JDBC ConnectorSource
- SparkConnectorSource estructura
- CatalogSource estructura
- CatalogSource Estructura de MySQL
- Estructura de PostgreSQL CatalogSource
- Estructura de OracleSQL CatalogSource
- Estructura de MicrosoftSQL ServerCatalogSource
- CatalogKinesisSource estructura
- DirectKinesisSource estructura
- KinesisStreamingSourceOptions estructura
- CatalogKafkaSource estructura
- DirectKafkaSource estructura
- KafkaStreamingSourceOptions estructura
- RedshiftSource estructura
- AmazonRedshiftSource estructura
- AmazonRedshiftNodeData estructura
- AmazonRedshiftAdvancedOption estructura
- Estructura de opción
- CatalogSource estructura S3
- SourceAdditionalOptions Estructura de S3
- CsvSource Estructura S3
- Estructura DirectJDBCSource
- DirectSourceAdditionalOptions Estructura S3
- JsonSource Estructura S3
- ParquetSource Estructura S3
- Estructura de S3 DeltaSource
- CatalogDeltaSource Estructura S3

- CatalogDeltaSource estructura
- HudiSource Estructura S3
- Estructura S3 CatalogHudiSource
- CatalogHudiSource estructura
- Estructura de DynamoDB CatalogSource
- RelationalCatalogSource estructura
- estructura JDBC ConnectorTarget
- SparkConnectorTarget estructura
- BasicCatalogTarget estructura
- CatalogTarget Estructura de MySQL
- Estructura de PostgreSQL CatalogTarget
- Estructura de OracleSQL CatalogTarget
- Estructura de Microsoft SQL ServerCatalogTarget
- RedshiftTarget estructura
- AmazonRedshiftTarget estructura
- UpsertRedshiftTargetOptions estructura
- CatalogTarget estructura S3
- GlueParquetTarget Estructura de S3
- CatalogSchemaChangePolicy estructura
- DirectTarget estructura S3
- HudiCatalogTarget Estructura S3
- Estructura S3 HudiDirectTarget
- Estructura S3 DeltaCatalogTarget
- DeltaDirectTarget Estructura S3
- DirectSchemaChangePolicy estructura
- ApplyMapping estructura
- Estructura de asignación
- SelectFields estructura
- DropFields estructura
- RenameField estructura

- Estructura Spigot
- Estructura Join
- JoinColumn estructura
- SplitFields estructura
- SelectFromCollection estructura
- FillMissingValues estructura
- Estructura Filter
- FilterExpression estructura
- FilterValue estructura
- CustomCode estructura
- Estructura SparkSQL
- SqlAlias estructura
- DropNullFields estructura
- NullCheckBoxList estructura
- NullValueField estructura
- Estructura Datatype
- Estructura Merge
- Estructura Union
- Estructura PIIDetection
- Estructura Aggregate
- DropDuplicates estructura
- GovernedCatalogTarget estructura
- GovernedCatalogSource estructura
- AggregateOperation estructura
- GlueSchema estructura
- GlueStudioSchemaColumn estructura
- GlueStudioColumn estructura
- DynamicTransform estructura
- TransformConfigParameter estructura
- EvaluateDataQuality estructura

- estructura DQ ResultsPublishingOptions
- Estructura DQ StopJobOnFailureOptions
- EvaluateDataQualityMultiFrame estructura
- Estructura de receta
- RecipeReference estructura
- SnowflakeNodeData estructura
- SnowflakeSource estructura
- SnowflakeTarget estructura
- ConnectorDataSource estructura
- ConnectorDataTarget estructura

- API de trabajos
  - Jobs
    - Tipos de datos
    - Estructura de trabajo
    - ExecutionProperty estructura
    - NotificationProperty estructura
    - JobCommand estructura
    - ConnectionsList estructura
    - JobUpdate estructura
    - SourceControlDetails estructura
    - Operaciones
    - CreateJob acción (Python: create_job)
    - UpdateJob acción (Python: update_job)
    - GetJob acción (Python: get_job)
    - GetJobs acción (Python: get_jobs)
    - DeleteJob acción (Python: delete_job)
    - ListJobs acción (Python: list_jobs)
    - BatchGetJobs acción (Python: batch_get_jobs)
  - Ejecuciones de trabajo
    - Tipos de datos

    - JobRun estructura
    - Estructura Predecessor
    - JobBookmarkEntry estructura
    - BatchStopJobRunSuccessfulSubmission estructura
    - BatchStopJobRunError estructura
    - NotificationProperty estructura
    - Operaciones
    - StartJobRun acción (Python: start_job_run)
    - BatchStopJobRun acción (Python: batch_stop_job_run)
    - GetJobRun acción (Python: get_job_run)
    - GetJobRuns acción (Python: get_job_runs)
    - GetJobBookmark acción (Python: get_job_bookmark)
    - GetJobBookmarks acción (Python: get_job_bookmarks)
    - ResetJobBookmark acción (Python: reset_job_bookmark)
- Desencadenadores
    - Tipos de datos
    - Estructura del desencadenador
    - Estructura de TriggerUpdate
    - Estructura de Predicate
    - Estructura de Condition
    - Estructura de acción
    - Estructura EventBatchingCondition
    - Operaciones
    - Acción CreateTrigger (Python: create_trigger)
    - Acción StartTrigger (Python: start_trigger)
    - Acción GetTrigger (Python: get_trigger)
    - Acción GetTriggers (Python: get_triggers)
    - Acción UpdateTrigger (Python: update_trigger)
    - Acción StopTrigger (Python: stop_trigger)
    - Acción DeleteTrigger (Python: delete_trigger)

    - Acción ListTriggers (Python: list_triggers)
    - Acción BatchGetTriggers (Python: batch_get_triggers)
- API de sesiones interactivas
    - Tipos de datos
    - Estructura de sesión
    - SessionCommand estructura
    - Estructura de instrucción
    - StatementOutput estructura
    - StatementOutputData estructura
    - ConnectionsList estructura
    - Operaciones
    - CreateSession acción (Python: create_session)
    - StopSession acción (Python: stop_session)
    - DeleteSession acción (Python: delete_session)
    - GetSession acción (Python: get_session)
    - ListSessions acción (Python: list_sessions)
    - RunStatement acción (Python: run_statement)
    - CancelStatement acción (Python: cancel_statement)
    - GetStatement acción (Python: get_statement)
    - ListStatements acción (Python: list_statements)
- API de puntos de conexión de desarrollo
    - Tipos de datos
    - Estructura DevEndpoint
    - Estructura DevEndpointCustomLibraries
    - Operaciones
    - Acción CreateDevEndpoint (Python: create_dev_endpoint)
    - Acción UpdateDevEndpoint (Python: update_dev_endpoint)
    - Acción DeleteDevEndpoint (Python: delete_dev_endpoint)
    - Acción GetDevEndpoint (Python: get_dev_endpoint)
    - Acción GetDevEndpoints (Python: get_dev_endpoints)

- Acción BatchGetDevEndpoints (Python: batch_get_dev_endpoints)
- Acción ListDevEndpoints (Python: list_dev_endpoints)

- Schema Registry
  - Tipos de datos
  - RegistryId estructura
  - RegistryListItem estructura
  - MetadataInfo estructura
  - OtherMetadataValueListItem estructura
  - SchemaListItem estructura
  - SchemaVersionListItem estructura
  - MetadataKeyValuePair estructura
  - SchemaVersionErrorItem estructura
  - ErrorDetails estructura
  - SchemaVersionNumber estructura
  - SchemaId estructura
  - Operaciones
  - CreateRegistry acción (Python: create_registry)
  - CreateSchema acción (Python: create_schema)
  - GetSchema acción (Python: get_schema)
  - ListSchemaVersions acción (Python: list_schema_versions)
  - GetSchemaVersion acción (Python: get_schema_version)
  - GetSchemaVersionsDiff acción (Python: get_schema_versions_diff)
  - ListRegistries acción (Python: list_registries)
  - ListSchemas acción (Python: list_schemas)
  - RegisterSchemaVersion acción (Python: register_schema_version)
  - UpdateSchema acción (Python: update_schema)
  - CheckSchemaVersionValidity acción (Python: check_schema_version_valid)
  - UpdateRegistry acción (Python: update_registry)
  - GetSchemaByDefinition acción (Python: get_schema_by_definition)
  - GetRegistry acción (Python: get_registry)

- PutSchemaVersionMetadata acción (Python: put_schema_version_metadata)
- QuerySchemaVersionMetadata acción (Python: query_schema_version_metadata)
- RemoveSchemaVersionMetadata acción (Python: remove_schema_version_metadata)
- DeleteRegistry acción (Python: delete_registry)
- DeleteSchema acción (Python: delete_schema)
- DeleteSchemaVersions acción (Python: delete_schema_versions)

- Flujos de trabajo
  - Tipos de datos
  - Estructura JobNodeDetails
  - Estructura CrawlerNodeDetails
  - Estructura TriggerNodeDetails
  - Estructura de rastreo
  - Estructura de nodos
  - Estructura perimetral
  - Estructura de flujo de trabajo
  - Estructura WorkflowGraph
  - Estructura WorkflowRun
  - Estructura WorkflowRunStatistics
  - Estructura StartingEventBatchCondition
  - Estructura de proyecto
  - Estructura BlueprintDetails
  - Estructura LastActiveDefinition
  - Estructura BlueprintRun
  - Operaciones
  - Acción CreateWorkflow (Python: create_workflow)
  - Acción UpdateWorkflow (Python: update_workflow)
  - DeleteWorkflow Action (Python: delete_workflow)
  - Acción GetWorkFlow (Python: get_workflow)
  - Acción ListWorkflows (Python: list_workflows)
  - Acción BatchGetWorkflows (Python: batch_get_workflows)

- Acción GetWorkflowRun (Python: get_workflow_run)
- Acción GetWorkflowRuns (Python: get_workflow_runs)
- Acción GetWorkflowRunProperties (Python: get_workflow_run_properties)
- PutWorkflowRunProperties Action (Python: put_workflow_run_properties)
- Acción CreateBlueprint (Python: create_blueprint)
- Acción UpdateBlueprint (Python: update_blueprint)
- Acción DeleteBlueprint (Python: delete_blueprint)
- Acción ListBlueprints (Python: list_blueprints)
- Acción BatchGetBlueprints (Python: batch_get_blueprints)
- Acción StartBlueprintRun (Python: start_blueprint_run)
- Acción GetBlueprintRun (Python: get_blueprint_run)
- Acción GetBlueprintRuns (Python: get_blueprint_runs)
- Acción StartWorkflowRun (Python: start_workflow_run)
- Acción StopWorkflowRun (Python: stop_workflow_run)
- Acción ResumeWorkflowRun (Python: resume_workflow_run)

- Perfiles de uso
  - Tipos de datos
  - ProfileConfiguration estructura
  - ConfigurationObject estructura
  - UsageProfileDefinition estructura
  - Operaciones
  - CreateUsageProfile acción (Python: create_usage_profile)
  - GetUsageProfile acción (Python: get_usage_profile)
  - UpdateUsageProfile acción (Python: update_usage_profile)
  - DeleteUsageProfile acción (Python: delete_usage_profile)
  - ListUsageProfiles acción (Python: list_usage_profiles)
- API de machine learning
  - Tipos de datos
  - Estructura TransformParameters
  - Estructura EvaluationMetrics

- Estructura MLTransform
- Estructura FindMatchesParameters
- Estructura FindMatchesMetrics
- Estructura ConfusionMatrix
- Estructura GlueTable
- Estructura TaskRun
- Estructura TransformFilterCriteria
- Estructura TransformSortCriteria
- Estructura TaskRunFilterCriteria
- Estructura TaskRunSortCriteria
- Estructura TaskRunProperties
- Estructura FindMatchesTaskRunProperties
- Estructura ImportLabelsTaskRunProperties
- Estructura ExportLabelsTaskRunProperties
- Estructura LabelingSetGenerationTaskRunProperties
- Estructura SchemaColumn
- Estructura TransformEncryption
- Estructura MLUserDataEncryption
- Estructura ColumnImportance
- Operaciones
- Acción CreateMLTransform (Python: create_ml_transform)
- Acción UpdateMLTransform (Python: update_ml_transform)
- Acción DeleteMLTransform (Python: delete_ml_transform)
- Acción GetMLTransform (Python: get_ml_transform)
- Acción GetMLTransforms (Python: get_ml_transforms)
- Acción ListMLTransforms (Python: list_ml_transforms)
- Acción StartMLEvaluationTaskRun (Python: start_ml_evaluation_task_run)
- Acción StartMLLabelingSetGenerationTaskRun (Python: start_ml_labeling_set_generation_task_run)

- Acción GetMLTaskRun (Python: get_ml_task_run)

- Acción GetMLTaskRuns (Python: get_ml_task_runs)
- Acción CancelMLTaskRun (Python: cancel_ml_task_run)
- Acción StartExportLabelsTaskRun (Python: start_export_labels_task_run)
- Acción StartImportLabelsTaskRun (Python: start_import_labels_task_run)

- API de Calidad de datos
  - Tipos de datos
  - DataSource estructura
  - DataQualityRulesetListDetails estructura
  - DataQualityTargetTable estructura
  - DataQualityRulesetEvaluationRunDescription estructura
  - DataQualityRulesetEvaluationRunFilter estructura
  - DataQualityEvaluationRunAdditionalRunOptions estructura
  - DataQualityRuleRecommendationRunDescription estructura
  - DataQualityRuleRecommendationRunFilter estructura
  - DataQualityResult estructura
  - DataQualityAnalyzerResult estructura
  - DataQualityObservation estructura
  - MetricBasedObservation estructura
  - DataQualityMetricValues estructura
  - DataQualityRuleResult estructura
  - DataQualityResultDescription estructura
  - DataQualityResultFilterCriteria estructura
  - DataQualityRulesetFilterCriteria estructura
  - Operaciones
  - StartDataQualityRulesetEvaluationRun acción (Python: start_data_quality_ruleset_evaluation_run)
  - CancelDataQualityRulesetEvaluationRun acción (Python: cancel_data_quality_ruleset_evaluation_run)
  - GetDataQualityRulesetEvaluationRun acción (Python: get_data_quality_ruleset_evaluation_run)
  - ListDataQualityRulesetEvaluationRuns acción (Python: list_data_quality_ruleset_evaluation_runs)

- StartDataQualityRuleRecommendationRun acción (Python: start_data_quality_rule_recommendation_run)
- CancelDataQualityRuleRecommendationRun acción (Python: cancel_data_quality_rule_recommendation_run)
- GetDataQualityRuleRecommendationRun acción (Python: get_data_quality_rule_recommendation_run)
- ListDataQualityRuleRecommendationRuns acción (Python: list_data_quality_rule_recommendation_runs)
- GetDataQualityResult acción (Python: get_data_quality_result)
- BatchGetDataQualityResult acción (Python: batch_get_data_quality_result)
- ListDataQualityResults acción (Python: list_data_quality_results)
- CreateDataQualityRuleset acción (Python: create_data_quality_ruleset)
- DeleteDataQualityRuleset acción (Python: delete_data_quality_ruleset)
- GetDataQualityRuleset acción (Python: get_data_quality_ruleset)
- ListDataQualityRulesets acción (Python: list_data_quality_rulesets)
- UpdateDataQualityRuleset acción (Python: update_data_quality_ruleset)

- API de detección de información confidencial
  - Tipos de datos
  - Estructura CustomEntityType
  - Operaciones
  - Acción CreateCustomEntityType (Python: create_custom_entity_type)
  - Acción DeleteCustomEntityType (Python: delete_custom_entity_type)
  - Acción GetCustomEntityType (Python: get_custom_entity_type)
  - Acción BatchGetCustomEntityTypes (Python: batch_get_custom_entity_types)
  - Acción ListCustomEntityTypes (Python: list_custom_entity_types)
- API de etiquetado en AWS Glue
  - Tipos de datos
  - Estructura de etiquetas
  - Operaciones
  - Acción TagResource (Python: tag_resource)
  - Acción UntagResource (Python: untag_resource)

- Acción GetTags (Python: get_tags)
- Tipos de datos comunes
  - Estructura de etiquetas
  - DecimalNumber estructura
  - ErrorDetail estructura
  - PropertyPredicate estructura
  - ResourceUri estructura
  - ColumnStatistics estructura
  - ColumnStatisticsError estructura
  - ColumnError estructura
  - ColumnStatisticsData estructura
  - BooleanColumnStatisticsData estructura
  - DateColumnStatisticsData estructura
  - DecimalColumnStatisticsData estructura
  - DoubleColumnStatisticsData estructura
  - LongColumnStatisticsData estructura
  - StringColumnStatisticsData estructura
  - BinaryColumnStatisticsData estructura
  - Patrones de cadena
- Excepciones
  - Estructura AccessDeniedException
  - Estructura de AlreadyExistsException
  - Estructura ConcurrentModificationException
  - Estructura ConcurrentRunsExceededException
  - Estructura CrawlerNotRunningException
  - Estructura CrawlerRunningException
  - Estructura CrawlerStoppingException
  - Estructura EntityNotFoundException
  - Estructura de FederationSourceException
  - Estructura de FederationSourceRetryableException

- Estructura GlueEncryptionException
- Estructura IdempotentParameterMismatchException
- Estructura IllegalWorkflowStateException
- Estructura InternalServiceException
- Estructura InvalidExecutionEngineException
- Estructura InvalidInputException
- Estructura InvalidStateException
- Estructura InvalidTaskStatusTransitionException
- Estructura JobDefinitionErrorException
- Estructura JobRunInTerminalStateException
- Estructura JobRunInvalidStateTransitionException
- Estructura JobRunNotInTerminalStateException
- Estructura LateRunnerException
- Estructura NoScheduleException
- Estructura OperationTimeoutException
- Estructura ResourceNotReadyException
- Estructura ResourceNumberLimitExceededException
- Estructura SchedulerNotRunningException
- Estructura SchedulerRunningException
- Estructura SchedulerTransitioningException
- Estructura UnrecognizedRunnerException
- Estructura ValidationException
- Estructura VersionMismatchException

API de seguridad en AWS Glue

La API de seguridad describe los tipos de datos de seguridad y la API relacionada con la seguridad en AWS Glue.

Tipos de datos

- DataCatalogEncryptionSettings structure

- Estructura EncryptionAtRest
- Estructura ConnectionPasswordEncryption
- Estructura EncryptionConfiguration
- Estructura S3Encryption
- Estructura CloudWatchEncryption
- Estructura JobBookmarksEncryption
- Estructura SecurityConfiguration
- Estructura GluePolicy

DataCatalogEncryptionSettings structure

Contiene información de configuración para mantener la seguridad del catálogo de datos.

Campos

- `EncryptionAtRest`: un objeto EncryptionAtRest.

  Especifica la configuración de cifrado en reposo del catálogo de datos.
- `ConnectionPasswordEncryption`: un objeto ConnectionPasswordEncryption.

  Cuando está habilitada la protección de la contraseña de conexión, el catálogo de datos utiliza una clave proporcionada por el cliente para cifrar la contraseña como parte de `CreateConnection` o `UpdateConnection`, y almacenarla en el campo `ENCRYPTED_PASSWORD` de las propiedades de conexión. Puede habilitar el cifrado de catálogo o cifrado de solo por contraseña.

Estructura EncryptionAtRest

Especifica la configuración de cifrado en reposo del catálogo de datos.

Campos

- `CatalogEncryptionMode` – Obligatorio: cadena UTF-8 (valores válidos: `DISABLED` | `SSE-KMS="SSEKMS"` | `SSE-KMS-WITH-SERVICE-ROLE="SSEKMSWITHSERVICEROLE"`).

  Modo de cifrado en reposo para cifrar los datos del catálogo de datos.
- `SseAwsKmsKeyId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

El ID de la clave AWS KMS que se debe usar para el cifrado en reposo.

- `CatalogEncryptionServiceRole`: cadena UTF-8 que coincide con el Custom string pattern #24.

  El rol que asume AWS Glue para cifrar y descifrar los objetos del Catálogo de datos en nombre de quien realiza la llamada.

Estructura ConnectionPasswordEncryption

La estructura de datos que utiliza el catálogo de datos para cifrar la contraseña como parte de `CreateConnection` o `UpdateConnection`, y almacenarla en el campo `ENCRYPTED_PASSWORD` de las propiedades de conexión. Puede habilitar el cifrado de catálogo o cifrado de solo por contraseña.

Cuando llega una solicitud `CreationConnection` que contiene una contraseña, el Data Catalog primero cifra la contraseña usando su clave AWS KMS. A continuación, vuelve a cifrar todo el objeto de conexión si también está activada el cifrado de catálogo.

Este cifrado requiere que configure los permisos de clave AWS KMS KMS para habilitar o restringir el acceso a la clave de contraseña de acuerdo con sus requisitos de seguridad. Por ejemplo, es posible que desee que solo los administradores tengan permiso para descifrar la clave de contraseña.

Campos

- `ReturnConnectionPasswordEncrypted` – Obligatorio: Booleano.

  Cuando la marca `ReturnConnectionPasswordEncrypted` se establece en “true”, las contraseñas permanecen cifradas en las respuestas de `GetConnection` y `GetConnections`. Este cifrado tiene efecto independientemente del cifrado del catálogo.

- `AwsKmsKeyId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Una clave AWS KMS que se utiliza para cifrar la contraseña de conexión.

  Si la protección de la contraseña de conexión está habilitada, el que realiza la llamada de `CreateConnection` y `UpdateConnection` necesita, al menos, permiso `kms:Encrypt` en la clave AWS KMS especificada para cifrar las contraseñas antes de almacenarlas en el Data Catalog.

Puede establecer el permiso de descifrado para habilitar o restringir el acceso a la clave de contraseña de acuerdo con sus requisitos de seguridad.

Estructura EncryptionConfiguration

Especifica una configuración de cifrado.

Campos

- `S3Encryption`: matriz de objetos S3Encryption.

  La configuración de cifrado de datos de Amazon Simple Storage Service (Amazon S3).
- `CloudWatchEncryption`: un objeto CloudWatchEncryption.

  La configuración de cifrado para Amazon CloudWatch.
- `JobBookmarksEncryption`: un objeto JobBookmarksEncryption.

  La configuración de cifrado para marcadores de trabajo.

Estructura S3Encryption

Especifica cómo se deben cifrar los datos de Amazon Simple Storage Service (Amazon S3).

Campos

- `S3EncryptionMode`: cadena UTF-8 (valores válidos: `DISABLED` | `SSE-KMS="SSEKMS"` | `SSE-S3="SSES3"`).

  El modo de cifrado que usar para datos de Amazon S3.
- `KmsKeyArn`: cadena UTF-8 que coincide con el Custom string pattern #25.

  El nombre de recurso de Amazon (ARN) de la clave de KMS que utilizar para cifrar los datos.

Estructura CloudWatchEncryption

Especifica cómo se deben cifrar los datos de Amazon CloudWatch.

Campos

- `CloudWatchEncryptionMode`: cadena UTF-8 (valores válidos: `DISABLED` | `SSE-KMS="SSEKMS"`).

  Modo de cifrado que usar para datos de CloudWatch.

- `KmsKeyArn`: cadena UTF-8 que coincide con el Custom string pattern #25.

  El nombre de recurso de Amazon (ARN) de la clave de KMS que utilizar para cifrar los datos.

Estructura JobBookmarksEncryption

Especifica cómo se deben cifrar los datos de los marcadores de trabajo.

Campos

- `JobBookmarksEncryptionMode`: cadena UTF-8 (valores válidos: `DISABLED` | `CSE-KMS="CSEKMS"`).

  Modo de cifrado que usar para datos de marcadores de trabajos.

- `KmsKeyArn`: cadena UTF-8 que coincide con el Custom string pattern #25.

  El nombre de recurso de Amazon (ARN) de la clave de KMS que utilizar para cifrar los datos.

Estructura SecurityConfiguration

Especifica una configuración de seguridad.

Campos

- `Name`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la configuración de seguridad.

- `CreatedTimeStamp`: marca temporal.

  El momento en que se creó esta configuración de seguridad.

- `EncryptionConfiguration`: un objeto EncryptionConfiguration.

La configuración de cifrado asociada con esta configuración de seguridad.

Estructura GluePolicy

Estructura para devolver una política de recursos.

Campos

- `PolicyInJson`: cadena UTF-8 de 2 bytes de largo como mínimo.

  Contiene el documento de política solicitado, en formato JSON.
- `PolicyHash`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Contiene el valor de hash asociado a esta política.
- `CreateTime`: marca temporal.

  La fecha y hora en la que se creó la política.
- `UpdateTime`: marca temporal.

  La fecha y hora en la que se actualizó la política por última vez.

Operaciones

- Acción GetDataCatalogEncryptionSettings (Python: get_data_catalog_encryption_settings)
- Acción PutDataCatalogEncryptionSettings (Python: put_data_catalog_encryption_settings)
- Acción PutResourcePolicy (Python: put_resource_policy)
- Acción GetResourcePolicy (Python: get_resource_policy)
- Acción DeleteResourcePolicy (Python: delete_resource_policy)
- Acción CreateSecurityConfiguration (Python: create_security_configuration)
- Acción DeleteSecurityConfiguration (Python: delete_security_configuration)
- Acción GetSecurityConfiguration (Python: get_security_configuration)
- Acción GetSecurityConfigurations (Python: get_security_configurations)
- Acción GetResourcePolicies (Python: get_resource_policies)

Acción GetDataCatalogEncryptionSettings (Python: get_data_catalog_encryption_settings)

Recupera la configuración de seguridad para un catálogo especificado.

Solicitud

- `CatalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID del catálogo de datos para el que recuperar la configuración de seguridad. Si no se proporciona ninguno, se usará de forma predeterminada el ID de cuenta de AWS.

Respuesta

- `DataCatalogEncryptionSettings`: un objeto DataCatalogEncryptionSettings.

  La configuración de seguridad solicitada.

Errores

- `InternalServiceException`
- `InvalidInputException`
- `OperationTimeoutException`

Acción PutDataCatalogEncryptionSettings (Python: put_data_catalog_encryption_settings)

Establece la configuración de seguridad para un catálogo especificado. Después de que se ha establecido la configuración, a partir de entonces, se aplicará el cifrado especificado a cada escritura de catálogos.

Solicitud

- `CatalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

El ID del catálogo de datos para el que establecer la configuración de seguridad. Si no se proporciona ninguno, se usará de forma predeterminada el ID de cuenta de AWS.

- `DataCatalogEncryptionSettings`: obligatorio: objeto DataCatalogEncryptionSettings.

  La configuración de seguridad que se va a establecer.

Respuesta

- Sin parámetros de respuesta.

Errores

- `InternalServiceException`
- `InvalidInputException`
- `OperationTimeoutException`

Acción PutResourcePolicy (Python: put_resource_policy)

Establece el catálogo de datos de política de recursos para el control de acceso.

Solicitud

- `PolicyInJson`: obligatorio: cadena UTF-8 de 2 bytes de largo como mínimo.

  Contiene el documento de política para establecer, en formato JSON.
- `ResourceArn`: cadena UTF-8, con 1 byte de largo como mínimo y 10 240 bytes de largo como máximo, que coincide con el Custom string pattern #22.

  No utilizar. Para uso interno únicamente.
- `PolicyHashCondition`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El valor hash devuelto cuando la política anterior se ha establecido mediante `PutResourcePolicy`. Su objetivo es evitar modificaciones simultáneas de una política. No utilice este parámetro si no se ha establecido ninguna política anterior.
- `PolicyExistsCondition`: cadena UTF-8 (valores válidos: `MUST_EXIST` | `NOT_EXIST` | `NONE`).

Un valor de `MUST_EXIST` se utiliza para actualizar una política. Un valor de `NOT_EXIST` se utiliza para crear una política nueva. Si se utiliza el valor de `NONE` o un valor nulo, la llamada no dependerá de la existencia de una política.

- `EnableHybrid`: cadena UTF-8 (valores válidos: `TRUE` | `FALSE`).

  Si se utiliza `'TRUE'`, significa que está utilizando ambos métodos para conceder acceso entre cuentas a recursos del Data Catalog:

  - Al actualizar directamente la política de recursos con `PutResourePolicy`
  - Al utilizar el comando Grant permissions (Concesión de permisos) en la AWS Management Console.

  Debe establecerse en `'TRUE'` si ya ha utilizado Management Console para conceder acceso entre cuentas; de lo contrario, se producirá un error en la llamada. La opción predeterminada es 'FALSO'.

Respuesta

- `PolicyHash`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Un hash de la política que se acaba de definir. Este debe incluirse en una llamada posterior que sobrescribe o actualiza esta política.

Errores

- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`
- `ConditionCheckFailureException`

Acción GetResourcePolicy (Python: get_resource_policy)

Recupera una política de recursos especificada.

Solicitud

- `ResourceArn`: cadena UTF-8, con 1 byte de largo como mínimo y 10 240 bytes de largo como máximo, que coincide con el Custom string pattern #22.

  El ARN del recurso de AWS Glue para el que se recuperará la política de recursos. Si no se proporciona, se devuelve la política de recursos del Data Catalog. Use `GetResourcePolicies` para ver todas las políticas de recursos existentes. Para obtener más información, consulte Especificación de ARN de recursos de AWS Glue.

Respuesta

- `PolicyInJson`: cadena UTF-8 de 2 bytes de largo como mínimo.

  Contiene el documento de política solicitado, en formato JSON.

- `PolicyHash`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Contiene el valor de hash asociado a esta política.

- `CreateTime`: marca temporal.

  La fecha y hora en la que se creó la política.

- `UpdateTime`: marca temporal.

  La fecha y hora en la que se actualizó la política por última vez.

Errores

- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`

Acción DeleteResourcePolicy (Python: delete_resource_policy)

Elimina una política especificada.

Solicitud

- `PolicyHashCondition`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El valor de hash devuelto cuando esta política se definió.
- `ResourceArn`: cadena UTF-8, con 1 byte de largo como mínimo y 10 240 bytes de largo como máximo, que coincide con el Custom string pattern #22.

  El ARN del recurso de AWS Glue para la política de recursos que se eliminará.

Respuesta

- Sin parámetros de respuesta.

Errores

- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`
- `ConditionCheckFailureException`

Acción CreateSecurityConfiguration (Python: create_security_configuration)

Crea una nueva configuración de seguridad. Una configuración de seguridad es un conjunto de propiedades de seguridad que AWS Glue puede usar. Puede utilizar una configuración de seguridad para cifrar los datos en reposo. Para obtener información acerca de cómo usar configuraciones de seguridad en AWS Glue, consulte Cifrado de datos escritos por rastreadores, trabajos y puntos de enlace de desarrollo.

Solicitud

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la nueva configuración de seguridad.

- `EncryptionConfiguration` – Obligatorio: objeto EncryptionConfiguration.

  La configuración de cifrado para la nueva configuración de seguridad.

Respuesta

- `Name`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre asignado a la nueva configuración de seguridad.
- `CreatedTimestamp`: marca temporal.

  El momento en que se creó la nueva configuración de seguridad.

Errores

- `AlreadyExistsException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`

Acción DeleteSecurityConfiguration (Python: delete_security_configuration)

Elimina una configuración de seguridad especificada.

Solicitud

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la configuración de seguridad que se va a eliminar.

Respuesta

- Sin parámetros de respuesta.

Errores

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

Acción GetSecurityConfiguration (Python: get_security_configuration)

Recupera una configuración de seguridad especificada.

Solicitud

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la configuración de seguridad que se va a recuperar.

Respuesta

- `SecurityConfiguration`: un objeto SecurityConfiguration.

  La configuración de seguridad solicitada.

Errores

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

Acción GetSecurityConfigurations (Python: get_security_configurations)

Recupera una lista de todas las configuraciones de seguridad.

Solicitud

- `MaxResults`: número (entero) que no es inferior a 1 ni es superior a 1000.

  El número máximo de resultados que devolver.
- `NextToken`: cadena UTF-8.

  Token de continuación si se trata de una llamada de continuidad.

Respuesta

- `SecurityConfigurations`: matriz de objetos SecurityConfiguration.

  Una lista de las configuraciones de seguridad.
- `NextToken`: cadena UTF-8.

  Un token de continuación, si hay más configuraciones de seguridad que devolver.

Errores

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

Acción GetResourcePolicies (Python: get_resource_policies)

Recupera las políticas de recursos configuradas en recursos individuales por AWS Resource Access Manager durante la concesión de permisos entre cuentas. También recupera el la política de recursos del Data Catalog.

Si ha habilitado el cifrado de metadatos en la configuración del Data Catalog y no tiene permiso en la clave AWS KMS, la operación no puede devolver la política de recursos del Data Catalog.

Solicitud

- `NextToken`: cadena UTF-8.

Token de continuación, si se trata de una solicitud de continuidad.

- `MaxResults`: número (entero) que no es inferior a 1 ni es superior a 1000.

  Tamaño máximo de una lista que se devolverá.

Respuesta

- `GetResourcePoliciesResponseList`: matriz de objetos Política de Glue.

  Una lista de las políticas de recursos individuales y la política de recursos a nivel de cuenta.

- `NextToken`: cadena UTF-8.

  Un token de continuación, si la lista devuelta no contiene la última política de recursos disponible.

Errores

- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`
- `GlueEncryptionException`

API del catálogo

La API del catálogo describe los tipos de datos y la API relacionada con el trabajo con catálogos en AWS Glue.

Temas

- API de la base de datos
- API de tabla
- API de partición
- API de conexión
- API de funciones definidas por el usuario
- Importación de un catálogo de Athena a AWS Glue

API de la base de datos

La API de la base de datos describe los tipos de datos de la base de datos e incluye la API de creación, eliminación, ubicación, actualización y enumeración de las bases de datos.

Tipos de datos

- Estructura de base de datos
- Estructura DatabaseInput
- Estructura PrincipalPermissions
- Estructura DataLakePrincipal
- Estructura DatabaseIdentifier
- Estructura de FederatedDatabase

Estructura de base de datos

El objeto `Database` representa una agrupación lógica de tablas que puede residir en un metaalmacén de Hive o en un RDBMS.

Campos

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la base de datos. Para su compatibilidad con Hive, este se incorpora en minúsculas al almacenarse.
- `Description`: cadena de descripción de un máximo de 2048 bytes de largo, que coincide con el URI address multi-line string pattern.

  Una descripción de la base de datos.
- `LocationUri`: identificador uniforme de recursos (uri), con 1 byte de largo como mínimo y 1024 bytes de largo como máximo, que coincide con el URI address multi-line string pattern.

  La ubicación de la base de datos (por ejemplo, una ruta de HDFS).
- `Parameters`: matriz de mapas de pares clave-valor.

  Cada clave es una cadena de claves con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

Cada valor es una cadena UTF-8 que no tiene más de 512000 bytes de largo.

Estos pares de clave-valor definen los parámetros y las propiedades de la base de datos.

- `CreateTime`: marca temporal.

  Hora de creación de la base de datos de metadatos en el catálogo.

- `CreateTableDefaultPermissions`: matriz de objetos PrincipalPermissions.

  Permite crear un conjunto de permisos predeterminados en la tabla para las entidades principales. Utilizado por AWS Lake Formation. No se utiliza en el transcurso normal de las operaciones de AWS Glue.

- `TargetDatabase`: un objeto DatabaseIdentifier.

  Estructura `DatabaseIdentifier` que describe una base de datos de destino para la vinculación de recursos.

- `CatalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID del Catálogo de datos donde reside la base de datos.

- `FederatedDatabase`: un objeto FederatedDatabase.

  Una estructura `FederatedDatabase` que hace referencia a una entidad externa aAWS Glue Data Catalog.

Estructura DatabaseInput

Estructura usada para crear o actualizar una base de datos.

Campos

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la base de datos. Para su compatibilidad con Hive, este se incorpora en minúsculas al almacenarse.

- `Description`: cadena de descripción de un máximo de 2048 bytes de largo, que coincide con el URI address multi-line string pattern.

Una descripción de la base de datos.

- `LocationUri`: identificador uniforme de recursos (uri), con 1 byte de largo como mínimo y 1024 bytes de largo como máximo, que coincide con el URI address multi-line string pattern.

  La ubicación de la base de datos (por ejemplo, una ruta de HDFS).

- `Parameters`: matriz de mapas de pares clave-valor.

  Cada clave es una cadena de claves con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Cada valor es una cadena UTF-8 que no tiene más de 512000 bytes de largo.

  Estos pares de clave-valor definen los parámetros y las propiedades de la base de datos.

  Estos pares de clave-valor definen los parámetros y las propiedades de la base de datos.

- `CreateTableDefaultPermissions`: matriz de objetos PrincipalPermissions.

  Permite crear un conjunto de permisos predeterminados en la tabla para las entidades principales. Utilizado por AWS Lake Formation. No se utiliza en el transcurso normal de las operaciones de AWS Glue.

- `TargetDatabase`: un objeto DatabaseIdentifier.

  Estructura `DatabaseIdentifier` que describe una base de datos de destino para la vinculación de recursos.

- `FederatedDatabase`: un objeto FederatedDatabase.

  Una estructura `FederatedDatabase` que hace referencia a una entidad externa aAWS Glue Data Catalog.

Estructura PrincipalPermissions

Permisos concedidos a una entidad principal.

Campos

- `Principal`: un objeto DataLakePrincipal.

  La entidad principal a la que se conceden los permisos.

- `Permissions`: matriz de cadenas UTF-8.

  Los permisos que se conceden al principal.

Estructura DataLakePrincipal

La AWS Lake Formation principal.

Campos

- `DataLakePrincipalIdentifier`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo.

  Un identificador para la AWS Lake Formation principal.

Estructura DatabaseIdentifier

Estructura que describe una base de datos de destino para la vinculación de recursos.

Campos

- `CatalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID del Catálogo de datos donde reside la base de datos.
- `DatabaseName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre de la base de datos del catálogo.
- `Region`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Región de la base de datos objetivo.

Estructura de FederatedDatabase

Una base de datos que apunta a una entidad externa a AWS Glue Data Catalog.

Campos

- `Identifier`: cadena UTF-8, con 1 byte de largo como mínimo y 512 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Un identificador único para la base de datos federada.
- `ConnectionName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la conexión al metalmacén externo.

Operaciones

- Acción CreateDatabase (Python: create_database)
- Acción UpdateDatabase (Python: update_database)
- Acción DeleteDatabase (Python: delete_database)
- Acción GetDatabase (Python: get_database)
- Acción GetDatabases (Python: get_databases)

Acción CreateDatabase (Python: create_database)

Crea una base de datos nueva en un Catálogo de datos.

Solicitud

- `CatalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID del Catálogo de datos donde se crea la base de datos. Si no se proporciona ninguno, se usará de forma predeterminada el ID de cuenta de AWS.
- `DatabaseInput`: obligatorio: objeto DatabaseInput.

  Los metadatos de la base de datos.
- `Tags`: matriz de mapas de pares de clave-valor, con 50 pares como máximo.

  Cada clave es una cadena UTF-8 con una longitud de entre 1 y 128 bytes.

  Cada valor es una cadena UTF-8 que no tiene más de 256 bytes de largo.

Las etiquetas que asigna a la base de datos.

Respuesta

- Sin parámetros de respuesta.

Errores

- `InvalidInputException`
- `AlreadyExistsException`
- `ResourceNumberLimitExceededException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`
- `ConcurrentModificationException`
- `FederatedResourceAlreadyExistsException`

Acción UpdateDatabase (Python: update_database)

Actualiza una definición de base de datos existente en un Catálogo de datos.

Solicitud

- `CatalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID del Catálogo de datos donde reside la base de datos de metadatos. Si no se proporciona ninguno, se usará de forma predeterminada el ID de cuenta de AWS.
- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre de la base de datos que debe actualizarse en el catálogo. Para su compatibilidad con Hive, se convierte en minúsculas.
- `DatabaseInput`: obligatorio: objeto DatabaseInput.

Objeto `DatabaseInput` que especifica la nueva definición de la base de datos de metadatos en el catálogo.

Respuesta

- Sin parámetros de respuesta.

Errores

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`
- `ConcurrentModificationException`

Acción DeleteDatabase (Python: delete_database)

Elimina una base de datos especificada de un Catálogo de datos.

> **Note**
>
> Después de completar esta operación, ya no tendrá acceso a las tablas (y todas las versiones y particiones de tabla que podrían pertenecer a las tablas) ni a las funciones definidas por el usuario en la base de datos eliminada. AWS Glue elimina estos recursos "huérfanos" de manera asíncrona en forma oportuna, a discreción del servicio.
> Para asegurarse de la eliminación inmediata de todos los recursos relacionados, antes de llamar a `DeleteDatabase`, use `DeleteTableVersion` o `BatchDeleteTableVersion`, `DeletePartition` o `BatchDeletePartition`, `DeleteUserDefinedFunction` y `DeleteTable` o `BatchDeleteTable` para eliminar todos los recursos que pertenezcan a la base de datos.

Solicitud

- `CatalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID del Catálogo de datos donde reside la base de datos. Si no se proporciona ninguno, se usará de forma predeterminada el ID de cuenta de AWS.
- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la base de datos que se va a eliminar. Para su compatibilidad con Hive, este debe estar completamente en minúsculas.

Respuesta

- Sin parámetros de respuesta.

Errores

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `ConcurrentModificationException`

Acción GetDatabase (Python: get_database)

Recupera la definición de una base de datos especificada.

Solicitud

- `CatalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID del Catálogo de datos donde reside la base de datos. Si no se proporciona ninguno, se usará de forma predeterminada el ID de cuenta de AWS.

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre de la base de datos que debe recuperarse. Para su compatibilidad con Hive, el nombre debe estar completamente en minúsculas.

Respuesta

- `Database`: un objeto Base de datos.

  La definición de la base de datos especificada en el catálogo.

Errores

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`
- `FederationSourceException`

Acción GetDatabases (Python: get_databases)

Recupera todas las bases de datos definidas en un Catálogo de datos determinado.

Solicitud

- `CatalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID del Catálogo de datos donde se recuperará `Databases`. Si no se proporciona ninguno, se usará de forma predeterminada el ID de cuenta de AWS.
- `NextToken`: cadena UTF-8.

  Token de continuación si se trata de una llamada de continuidad.
- `MaxResults`: número (entero) que no es inferior a 1 ni es superior a 100.

Número máximo de bases de datos que se devuelven en una respuesta.

- `ResourceShareType`: cadena UTF-8 (valores válidos: `FOREIGN` | `ALL` | `FEDERATED`).

  Le permite especificar que desea enumerar las bases de datos compartidas con su cuenta. Los valores permitidos son `FEDERATED`, `FOREIGN` o `ALL`.

  - Si se establece en `FEDERATED`, enumerará las bases de datos federadas (que hacen referencia a una entidad externa) compartidas con la cuenta.
  - Si se establece en `FOREIGN`, enumerará las bases de datos compartidas con su cuenta.
  - Si se establece en `ALL`, enumerará las bases de datos compartidas con su cuenta, así como las bases de datos de su cuenta local.

Respuesta

- `DatabaseList` (obligatorio): una matriz de objetos Base de datos.

  Lista de objetos `Database` del catálogo especificado.

- `NextToken`: cadena UTF-8.

  Token de continuación para paginar la lista de tokens obtenida; se devuelve si el segmento actual de la lista no es el último.

Errores

- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`

API de tabla

La API de tabla describe los tipos de datos y las operaciones que se asocian a las tablas.

Tipos de datos

- Estructura de tabla

- TableInput estructura
- FederatedTable estructura
- Estructura de las columnas
- StorageDescriptor estructura
- SchemaReference estructura
- SerDeInfo estructura
- Estructura Order
- SkewedInfo estructura
- TableVersion estructura
- TableError estructura
- TableVersionError estructura
- SortCriterion estructura
- TableIdentifier estructura
- KeySchemaElement estructura
- PartitionIndex estructura
- PartitionIndexDescriptor estructura
- BackfillError estructura
- IcebergInput estructura
- OpenTableFormatInput estructura
- ViewDefinition estructura
- ViewDefinitionInput estructura
- ViewRepresentation estructura
- ViewRepresentationInput estructura

Estructura de tabla

Representa una recopilación de datos relacionados organizados en columnas y filas.

Campos

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

El nombre de la tabla. Para su compatibilidad con Hive, este debe estar completamente en minúsculas.

- `DatabaseName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la base de datos de metadatos donde residen los metadatos de la tabla. Para su compatibilidad con Hive, este debe estar completamente en minúsculas.

- `Description`: cadena de descripción de un máximo de 2048 bytes de largo, que coincide con el URI address multi-line string pattern.

  Una descripción de la tabla.

- `Owner`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El propietario de la tabla.

- `CreateTime`: marca temporal.

  La hora de creación de la definición de la tabla en el Catálogo de datos.

- `UpdateTime`: marca temporal.

  La última vez que se actualizó la tabla.

- `LastAccessTime`: marca temporal.

  La última vez que se accedió a tabla. Esta suele tomarse de HDFS y podría no ser de confianza.

- `LastAnalyzedTime`: marca temporal.

  La última vez que se calcularon las estadísticas de columna para esta tabla.

- `Retention`: número (entero), cero como máximo.

  El tiempo de retención para esta tabla.

- `StorageDescriptor`: un objeto StorageDescriptor.

  Un descriptor de almacenamiento que contiene información acerca del almacenamiento físico de esta tabla.

- `PartitionKeys`: matriz de objetos Columna.

Una lista de columnas por las que se particiona la tabla. Solo se admiten los tipos primitivos como claves de partición.

Cuando se crea una tabla utilizada por Amazon Athena, y no especifica ninguna `partitionKeys`, al menos debe establecer el valor de `partitionKeys` en una lista vacía. Por ejemplo:

```
"PartitionKeys": []
```

- `ViewOriginalText`: cadena UTF-8 de 409600 bytes de largo como máximo.

  Se incluye para permitir la compatibilidad con Apache Hive. No se utiliza en el curso normal de AWS Glue las operaciones. Si la tabla es una `VIRTUAL_VIEW` Athena configuración determinada codificada en base64.

- `ViewExpandedText`: cadena UTF-8 de 409600 bytes de largo como máximo.

  Se incluye para permitir la compatibilidad con Apache Hive. No se utiliza en el curso normal de AWS Glue las operaciones.

- `TableType`: cadena UTF-8 de 255 bytes de largo como máximo.

  El tipo de esta tabla. AWS Glue creará tablas con `EXTERNAL_TABLE` este tipo. Otros servicios, por ejemplo Athena, pueden crear tablas con tipos de tablas adicionales.

  AWS Glue tipos de tablas relacionadas:

  EXTERNAL_TABLE

  Atributo de compatibilidad con Hive: indica una tabla no gestionada por Hive.

  GOVERNED

  Utilizado por AWS Lake Formation. El catálogo AWS Glue de datos entiende`GOVERNED`.

- `Parameters`: matriz de mapas de pares clave-valor.

  Cada clave es una cadena de claves con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Cada valor es una cadena UTF-8 que no tiene más de 512000 bytes de largo.

  Estos pares clave-valor definen las propiedades asociadas a la tabla.

- `CreatedBy`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

La persona o entidad que creó la tabla.

- `IsRegisteredWithLakeFormation`: booleano.

  Indica si la tabla se ha registrado con AWS Lake Formation.

- `TargetTable`: un objeto TableIdentifier.

  Estructura `TableIdentifier` que describe una tabla de destino para la vinculación de recursos.

- `CatalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  ID del Catálogo de datos donde reside la tabla.

- `VersionId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID de la versión de la tabla.

- `FederatedTable`: un objeto FederatedTable.

  Una estructura `FederatedTable` que hace referencia a una entidad externa a AWS Glue Data Catalog.

- `ViewDefinition`: un objeto ViewDefinition.

  Estructura que contiene toda la información que define la vista, incluidos el dialecto o los dialectos de la vista, y la consulta.

- `IsMultiDialectView`: booleano.

  Especifica si la vista admite los dialectos SQL de uno o varios motores de consulta diferentes y, por lo tanto, si esos motores pueden leerla.

TableInput estructura

Una estructura utilizada para definir una tabla.

Campos

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

El nombre de la tabla. Para su compatibilidad con Hive, este se incorpora en minúsculas al almacenarse.

- `Description`: cadena de descripción de un máximo de 2048 bytes de largo, que coincide con el URI address multi-line string pattern.

  Una descripción de la tabla.

- `Owner`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El propietario de la tabla. Se incluye para permitir la compatibilidad con Apache Hive. No se utiliza en el curso normal de AWS Glue las operaciones.

- `LastAccessTime`: marca temporal.

  La última vez que se accedió a tabla.

- `LastAnalyzedTime`: marca temporal.

  La última vez que se calcularon las estadísticas de columna para esta tabla.

- `Retention`: número (entero), cero como máximo.

  El tiempo de retención para esta tabla.

- `StorageDescriptor`: un objeto StorageDescriptor.

  Un descriptor de almacenamiento que contiene información acerca del almacenamiento físico de esta tabla.

- `PartitionKeys`: matriz de objetos Columna.

  Una lista de columnas por las que se particiona la tabla. Solo se admiten los tipos primitivos como claves de partición.

  Cuando se crea una tabla utilizada por Amazon Athena, y no especifica ninguna `partitionKeys`, al menos debe establecer el valor de `partitionKeys` en una lista vacía. Por ejemplo:

  ```
  "PartitionKeys": []
  ```

- `ViewOriginalText`: cadena UTF-8 de 409600 bytes de largo como máximo.

Se incluye para permitir la compatibilidad con Apache Hive. No se utiliza en el curso normal de AWS Glue las operaciones. Si la tabla es una `VIRTUAL_VIEW` Athena configuración determinada codificada en base64.

- `ViewExpandedText`: cadena UTF-8 de 409600 bytes de largo como máximo.

  Se incluye para permitir la compatibilidad con Apache Hive. No se utiliza en el curso normal de AWS Glue las operaciones.

- `TableType`: cadena UTF-8 de 255 bytes de largo como máximo.

  El tipo de esta tabla. AWS Glue creará tablas con `EXTERNAL_TABLE` este tipo. Otros servicios, por ejemplo Athena, pueden crear tablas con tipos de tablas adicionales.

  AWS Glue tipos de tablas relacionadas:

  EXTERNAL_TABLE

  Atributo de compatibilidad con Hive: indica una tabla no gestionada por Hive.

  GOVERNED

  Utilizado por AWS Lake Formation. El catálogo AWS Glue de datos entiende`GOVERNED`.

- `Parameters`: matriz de mapas de pares clave-valor.

  Cada clave es una cadena de claves con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Cada valor es una cadena UTF-8 que no tiene más de 512000 bytes de largo.

  Estos pares clave-valor definen las propiedades asociadas a la tabla.

- `TargetTable`: un objeto TableIdentifier.

  Estructura `TableIdentifier` que describe una tabla de destino para la vinculación de recursos.

- `ViewDefinition`: un objeto ViewDefinitionInput.

  Estructura que contiene toda la información que define la vista, incluidos el dialecto o los dialectos de la vista, y la consulta.

FederatedTable estructura

Una tabla que apunta a una entidad externa a AWS Glue Data Catalog.

Campos

- `Identifier`: cadena UTF-8, con 1 byte de largo como mínimo y 512 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Un identificador único para la tabla federada.
- `DatabaseIdentifier`: cadena UTF-8, con 1 byte de largo como mínimo y 512 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Un identificador único para la base de datos federada.
- `ConnectionName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la conexión al metalmacén externo.

Estructura de las columnas

Una columna en una `Table`.

Campos

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre del elemento `Column`.
- `Type`: cadena UTF-8 con un máximo de 131072 bytes de largo, que coincide con el Single-line string pattern.

  El tipo de datos de la `Column`.
- `Comment`: cadena de comentario de un máximo de 255 bytes de largo, que coincide con el Single-line string pattern.

  Un comentario de texto de formato libre.
- `Parameters`: matriz de mapas de pares clave-valor.

  Cada clave es una cadena de claves con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Cada valor es una cadena UTF-8 que no tiene más de 512000 bytes de largo.

Estos pares clave-valor. definen las propiedades asociadas a la columna.

StorageDescriptor estructura

Describe el almacenamiento físico de los datos de la tabla.

Campos

- `Columns`: matriz de objetos Columna.

  Una lista de las `Columns` de la tabla.
- `Location`: cadena de ubicación de un máximo de 2056 bytes de largo, que coincide con el URI address multi-line string pattern.

  Ubicación física de la tabla. De forma predeterminada, adopta la forma de la ubicación de almacén, seguida de la ubicación de la base de datos en el almacén, seguida del nombre de tabla.
- `AdditionalLocations`: matriz de cadenas UTF-8.

  Lista de ubicaciones que apuntan a la ruta en la que se encuentra una tabla Delta.
- `InputFormat`: cadena de formato de un máximo de 128 bytes de largo, que coincide con el Single-line string pattern.

  Formato de entrada: `SequenceFileInputFormat` (binario), `TextInputFormat` o formato personalizado.
- `OutputFormat`: cadena de formato de un máximo de 128 bytes de largo, que coincide con el Single-line string pattern.

  Formato de salida: `SequenceFileOutputFormat` (binario), `IgnoreKeyTextOutputFormat` o formato personalizado.
- `Compressed`: booleano.

  `True` si los datos de la tabla están comprimidos o `False` si no.
- `NumberOfBuckets`: número (entero).

  Debe especificarse si la tabla contiene alguna columna de dimensión.
- `SerdeInfo`: un objeto SerDeInfo.

  La información de serialización/deserialización (). SerDe

- `BucketColumns`: matriz de cadenas UTF-8.

  Lista de columnas de agrupamiento del reductor, columnas de clústeres y columnas de almacenamiento en bucket de la tabla.

- `SortColumns`: matriz de objetos Order.

  Una lista donde se especifica el orden de clasificación de cada bucket en la tabla.

- `Parameters`: matriz de mapas de pares clave-valor.

  Cada clave es una cadena de claves con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Cada valor es una cadena UTF-8 que no tiene más de 512000 bytes de largo.

  Las propiedades facilitadas por el usuario en formato clave-valor.

- `SkewedInfo`: un objeto SkewedInfo.

  La información sobre los valores que aparecen con frecuencia en una columna (valores sesgados).

- `StoredAsSubDirectories`: booleano.

  `True` si los datos de la tabla se almacenan en subdirectorios o `False` si no.

- `SchemaReference`: un objeto SchemaReference.

  Objeto que hace referencia a un esquema almacenado en el registro de esquemas. AWS Glue

  Al crear una tabla, puede pasar una lista vacía de columnas para el esquema y, en su lugar, utilizar una referencia de esquema.

SchemaReference estructura

Objeto que hace referencia a un esquema almacenado en el registro de AWS Glue esquemas.

Campos

- `SchemaId`: un objeto SchemaId.

  Estructura que contiene campos de identidad de esquema. Este o el `SchemaVersionId` tiene que ser proporcionado.

- `SchemaVersionId`: cadena UTF-8, con 36 bytes de largo como mínimo y 36 bytes de largo como máximo, que coincide con el Custom string pattern #17.

  El ID único asignado a una versión del esquema. Este o el `SchemaId` tiene que ser proporcionado.
- `SchemaVersionNumber`: número (largo) que no es inferior a 1 ni es superior a 100 000.

  El número de versión del esquema.

SerDeInfo estructura

Información sobre un programa de serialización/deserialización (SerDe) que sirve como extractor y cargador.

Campos

- `Name`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  SerDeNombre del.
- `SerializationLibrary`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Por lo general, la clase que implementa el SerDe. Un ejemplo es `org.apache.hadoop.hive.serde2.columnar.ColumnarSerDe`.
- `Parameters`: matriz de mapas de pares clave-valor.

  Cada clave es una cadena de claves con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Cada valor es una cadena UTF-8 que no tiene más de 512000 bytes de largo.

  Estos pares clave-valor definen los parámetros de inicialización para. SerDe

Estructura Order

Especifica el orden de clasificación de una columna ordenada.

Campos

- `Column`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la columna.
- `SortOrder` – Obligatorio: número (entero), uno como máximo.

  Indica que la columna se clasifica en orden ascendente (`== 1`) o descendente (`==0`).

SkewedInfo estructura

Especifica valores sesgados en una tabla. Los valores sesgados son los que se producen con una frecuencia muy alta.

Campos

- `SkewedColumnNames`: matriz de cadenas UTF-8.

  Una lista de nombres de columnas que contienen valores de sesgado.
- `SkewedColumnValues`: matriz de cadenas UTF-8.

  Una lista de valores que aparecen con tanta frecuencia como para considerarse de sesgado.
- `SkewedColumnValueLocationMaps`: matriz de mapas de pares clave-valor.

  Cada clave es una cadena UTF-8.

  Cada valor es una cadena UTF-8.

  Un mapeo de valores de sesgado a las columnas que los contienen.

TableVersion estructura

Especifica una versión de una tabla.

Campos

- `Table`: un objeto Tabla.

  La tabla en cuestión.

- `VersionId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El valor de ID que identifica la versión de esta tabla. Una `VersionId` es una representación de cadena de un número entero. Cada versión se incrementa en 1.

TableError estructura

Un registro de error para las operaciones con tablas.

Campos

- `TableName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la tabla. Para su compatibilidad con Hive, este debe estar completamente en minúsculas.
- `ErrorDetail`: un objeto ErrorDetail.

  Los detalles sobre el error.

TableVersionError estructura

Un registro de error para las operaciones con versiones de tablas.

Campos

- `TableName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre de la tabla en cuestión.
- `VersionId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El valor de ID de la versión en cuestión. Una `VersionID` es una representación de cadena de un número entero. Cada versión se incrementa en 1.
- `ErrorDetail`: un objeto ErrorDetail.

  Los detalles sobre el error.

SortCriterion estructura

Especifica un campo por el que se va a ordenar y un orden de clasificación.

Campos

- `FieldName`: cadena de valor, de 1024 bytes de largo como máximo.

  El nombre del campo en el que se va a ordenar.
- `Sort`: cadena UTF-8 (valores válidos: `ASC="ASCENDING"` | `DESC="DESCENDING"`).

  Orden ascendente o descendente.

TableIdentifier estructura

Estructura que describe una tabla de destino para la vinculación de recursos.

Campos

- `CatalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  ID del Catálogo de datos donde reside la tabla.
- `DatabaseName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre de la base de datos del catálogo que contiene la tabla de destino.
- `Name`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la tabla de destino.
- `Region`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Región de la tabla objetivo.

KeySchemaElement estructura

Un par de claves de partición que consta de un nombre y un tipo.

Campos

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de una clave de partición.

- `Type` – Obligatorio: cadena UTF-8, de 131 072 bytes de largo como máximo, que coincide con Single-line string pattern.

  El tipo de una clave de partición.

PartitionIndex estructura

Una estructura para un índice de partición.

Campos

- `Keys`: obligatorio: matriz de cadenas UTF-8, con una cadena como mínimo.

  Las claves para el índice de partición.

- `IndexName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre del índice de partición.

PartitionIndexDescriptor estructura

Un descriptor para un índice de partición en una tabla.

Campos

- `IndexName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre del índice de partición.

- `Keys` – Obligatorio: una matriz de objetos KeySchemaElement, 1 estructura como mínimo.

  Una lista de una o más claves, como estructuras `KeySchemaElement`, para el índice de partición.

- `IndexStatus` – Obligatorio: cadena UTF-8 (valores válidos: `CREATING` | `ACTIVE` | `DELETING` | `FAILED`).

  El estado del índice de partición.

  Los posibles estados son:

  - CREACIÓN: el índice se está creando. Cuando el índice está en un estado CREACIÓN, el índice o su tabla no se pueden eliminar.
  - ACTIVO: la creación del índice se realiza correctamente.
  - ERROR: error en la creación del índice.
  - ELIMINACIÓN: el índice se elimina de la lista de índices.
- `BackfillErrors`: matriz de objetos BackfillError.

  Una lista de los errores que pueden producirse al registrar índices de particiones para una tabla existente.

BackfillError estructura

Una lista de los errores que pueden producirse al registrar índices de particiones para una tabla existente.

Estos errores dan los detalles acerca de por qué un registro de índice presentó error y proporcionan un número limitado de particiones en la respuesta, de modo que pueda corregir las particiones con errores e intentar registrar el índice de nuevo. El conjunto más común de errores que pueden ocurrir se clasifican de la siguiente manera:

- EncryptedPartitionError: Las particiones están cifradas.
- InvalidPartitionTypeDataError: El valor de la partición no coincide con el tipo de datos de esa columna de partición.
- MissingPartitionValueError: Las particiones están cifradas.
- UnsupportedPartitionCharacterError: No se admiten los caracteres incluidos en el valor de la partición. Por ejemplo: U+0000, U+0001, U+0002.
- InternalError: cualquier error que no pertenezca a otros códigos de error.

Campos

- `Code`: cadena UTF-8 (valores válidos: `ENCRYPTED_PARTITION_ERROR` | `INTERNAL_ERROR` | `INVALID_PARTITION_TYPE_DATA_ERROR` | `MISSING_PARTITION_VALUE_ERROR` | `UNSUPPORTED_PARTITION_CHARACTER_ERROR`).

  Una lista de los errores que ocurrieron al registrar índices de particiones para una tabla existente.

- `Partitions`: matriz de objetos PartitionValueList.

  Una lista de un número limitado de particiones en la respuesta.

IcebergInput estructura

Una estructura que define una tabla de metadatos de Apache Iceberg para crearla en el catálogo.

Campos

- `MetadataOperation` – Obligatorio: cadena UTF-8 (valores válidos: `CREATE`).

  Una operación de metadatos obligatoria. Esto solo se puede configurar en `CREATE`.

- `Version`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  La versión de tabla para la tabla de Iceberg. El valor predeterminado es 2.

OpenTableFormatInput estructura

Una estructura que representa una tabla de formato abierto.

Campos

- `IcebergInput`: un objeto IcebergInput.

  Especifica una estructura `IcebergInput` que define una tabla de metadatos de Apache Iceberg.

ViewDefinition estructura

Estructura que contiene detalles para las representaciones.

Campos

- `IsProtected`: booleano.

  Puede establecer este indicador como verdadero para indicar al motor que no inserte las operaciones proporcionadas por el usuario en el plan lógico de la vista durante la planificación de las consultas. Sin embargo, establecer este indicador no garantiza que el motor cumpla con las normas. Consulte la documentación del motor para comprender las garantías que se ofrecen si las hubiera.

- `Definer`: cadena UTF-8, con 1 byte de largo como mínimo y 512 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El definidor de una vista en SQL.

- `SubObjects`: matriz de cadenas UTF-8, con 10 cadenas como máximo.

  Una lista de los nombres de recursos de Amazon (ARN) de la tabla.

- `Representations`: matriz de objetos ViewRepresentation, con una estructura como mínimo y 1000 estructuras como máximo.

  Una lista de representaciones.

ViewDefinitionInput estructura

Estructura que contiene detalles para crear o actualizar una AWS Glue vista.

Campos

- `IsProtected`: booleano.

  Puede establecer este indicador como verdadero para indicar al motor que no inserte las operaciones proporcionadas por el usuario en el plan lógico de la vista durante la planificación de las consultas. Sin embargo, establecer este indicador no garantiza que el motor cumpla con las normas. Consulte la documentación del motor para comprender las garantías que se ofrecen si las hubiera.

- `Definer`: cadena UTF-8, con 1 byte de largo como mínimo y 512 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El definidor de una vista en SQL.

- `Representations`: Una matriz de objetos ViewRepresentationInput, con 1 estructura como mínimo y 10 como máximo.

  Lista de estructuras que contiene el dialecto de la vista y la consulta que define la vista.
- `SubObjects`: matriz de cadenas UTF-8, con 10 cadenas como máximo.

  Una lista de los ARN de la tabla base que componen la vista.

ViewRepresentation estructura

Estructura que contiene el dialecto de la vista y la consulta que define la vista.

Campos

- `Dialect`: cadena UTF-8 (valores válidos: `REDSHIFT` | `ATHENA` | `SPARK`).

  El dialecto del motor de consultas.
- `DialectVersion`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo.

  La versión del dialecto del motor de consultas. Por ejemplo, 3.0.0.
- `ViewOriginalText`: cadena UTF-8 de 409600 bytes de largo como máximo.

  La consulta `SELECT` proporcionada por el cliente durante `CREATE VIEW DDL`. Este SQL no se usa durante una consulta en una vista (en su lugar, se usa `ViewExpandedText`). `ViewOriginalText` se usa en casos como `SHOW CREATE VIEW`, en los que los usuarios desean ver el comando DDL original que creó la vista.
- `ViewExpandedText`: cadena UTF-8 de 409600 bytes de largo como máximo.

  El SQL expandido para la vista. Los motores utilizan este SQL para procesar una consulta en una vista. Es posible que los motores hagan operaciones durante la creación de la vista para transformar `ViewOriginalText` en `ViewExpandedText`. Por ejemplo:
  - Identificadores totalmente cualificados: `SELECT * from table1 -> SELECT * from db1.table1`
- `ValidationConnection`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre de la conexión que se usará para validar la representación específica de la vista.

- `IsStale`: booleano.

  Los dialectos marcados como obsoletos ya no son válidos y deben actualizarse para poder consultarlos en sus respectivos motores de consulta.

ViewRepresentationInput estructura

Estructura que contiene detalles de una representación para actualizar o crear una vista de Lake Formation.

Campos

- `Dialect`: cadena UTF-8 (valores válidos: `REDSHIFT` | `ATHENA` | `SPARK`).

  Parámetro que especifica el tipo de motor de una representación específica.
- `DialectVersion`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo.

  Un parámetro que especifica la versión del motor de una representación específica.
- `ViewOriginalText`: cadena UTF-8 de 409600 bytes de largo como máximo.

  Una cadena que representa la consulta SQL original que describe la vista.
- `ValidationConnection`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre de la conexión que se usará para validar la representación específica de la vista.
- `ViewExpandedText`: cadena UTF-8 de 409600 bytes de largo como máximo.

  Cadena que representa la consulta SQL que describe la vista con los ARN de recursos ampliados.

Operaciones

- CreateTable acción (Python: create_table)
- UpdateTable acción (Python: update_table)
- DeleteTable acción (Python: delete_table)
- BatchDeleteTable acción (Python: batch_delete_table)
- GetTable acción (Python: get_table)

- GetTables acción (Python: get_tables)
- GetTableVersion acción (Python: get_table_version)
- GetTableVersions acción (Python: get_table_versions)
- DeleteTableVersion acción (Python: delete_table_version)
- BatchDeleteTableVersion acción (Python: batch_delete_table_version)
- SearchTables acción (Python: search_tables)
- GetPartitionIndexes acción (Python: get_partition_indexes)
- CreatePartitionIndex acción (Python: create_partition_index)
- DeletePartitionIndex acción (Python: delete_partition_index)
- GetColumnStatisticsForTable acción (Python: get_column_statistics_for_table)
- UpdateColumnStatisticsForTable acción (Python: update_column_statistics_for_table)
- DeleteColumnStatisticsForTable acción (Python: delete_column_statistics_for_table)

CreateTable acción (Python: create_table)

Crea una nueva definición de tabla en el Catálogo de datos.

Solicitud

- `CatalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID del Catálogo de datos donde se crea la `Table`. Si no se proporciona ninguno, se utiliza el identificador de AWS cuenta de forma predeterminada.
- `DatabaseName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  La base de datos de catálogos en la que se crea la nueva tabla. Para compatibilidad con Hive, este nombre está completamente en minúsculas.
- `TableInput`: obligatorio: objeto TableInput.

  Objeto `TableInput` que define la tabla de metadatos que se va a crear en el catálogo.
- `PartitionIndexes`: matriz de objetos PartitionIndex, con 3 estructuras como máximo.

  Una lista de índices de particiones, estructuras `PartitionIndex`, para crear en la tabla.

- `TransactionId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Custom string pattern #16.

  El ID de la transacción.

- `OpenTableFormatInput`: un objeto OpenTableFormatInput.

  Especifica una estructura `OpenTableFormatInput` al crear una tabla de formato abierto.

Respuesta

- Sin parámetros de respuesta.

Errores

- `AlreadyExistsException`
- `InvalidInputException`
- `EntityNotFoundException`
- `ResourceNumberLimitExceededException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`
- `ConcurrentModificationException`
- `ResourceNotReadyException`

UpdateTable acción (Python: update_table)

Actualiza una tabla de metadatos en el Catálogo de datos.

Solicitud

- `CatalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID del Catálogo de datos donde reside la tabla. Si no se proporciona ninguno, se utiliza el ID de la AWS cuenta de forma predeterminada.

- `DatabaseName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la base de datos de catálogos donde reside la tabla. Para compatibilidad con Hive, este nombre está completamente en minúsculas.

- `TableInput`: obligatorio: objeto TableInput.

  Objeto `TableInput` actualizado para definir la tabla de metadatos en el catálogo.

- `SkipArchive`: booleano.

  De forma predeterminada, `UpdateTable` crea siempre una versión archivada de la tabla antes de actualizarla. Si `skipArchive` se establece en true, sin embargo, `UpdateTable` no crea la versión archivada.

- `TransactionId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Custom string pattern #16.

  ID de transacción en el que se va a actualizar el contenido de la tabla.

- `VersionId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID de la versión a la que se actualizará el contenido de la tabla.

- `ViewUpdateAction`: cadena UTF-8 (valores válidos: `ADD` | `REPLACE` | `ADD_OR_REPLACE` | `DROP`).

  La operación que se llevará a cabo al actualizar la vista.

- `Force`: booleano.

  Un indicador que se puede establecer como verdadero para ignorar los requisitos de coincidencia entre el descriptor de almacenamiento y el subobjeto.

Respuesta

- Sin parámetros de respuesta.

Errores

- `EntityNotFoundException`

- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `ConcurrentModificationException`
- `ResourceNumberLimitExceededException`
- `GlueEncryptionException`
- `ResourceNotReadyException`

DeleteTable acción (Python: delete_table)

Quita una definición de tabla del Catálogo de datos.

> **Note**
>
> Después de completar esta operación, ya no tendrá acceso a las versiones de la tabla y a las particiones que pertenecen a la tabla. AWS Glue elimina estos recursos "huérfanos" de manera asíncrona en forma oportuna, a discreción del servicio.
> Para asegurarse de la eliminación inmediata de todos los recursos relacionados, antes de llamar a `DeleteTable`, use `DeleteTableVersion` o `BatchDeleteTableVersion` y `DeletePartition` o `BatchDeletePartition`, para eliminar todos los recursos que pertenezcan a la tabla.

Solicitud

- `CatalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID del Catálogo de datos donde reside la tabla. Si no se proporciona ninguno, se utiliza el ID de AWS cuenta de forma predeterminada.

- `DatabaseName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la base de datos de catálogos donde reside la tabla. Para compatibilidad con Hive, este nombre está completamente en minúsculas.

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la tabla que se eliminará. Para compatibilidad con Hive, este nombre está completamente en minúsculas.
- `TransactionId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Custom string pattern #16.

  ID de transacción en el que se va a eliminar el contenido de la tabla.

Respuesta

- Sin parámetros de respuesta.

Errores

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `ConcurrentModificationException`
- `ResourceNotReadyException`

BatchDeleteTable acción (Python: batch_delete_table)

Elimina varias tablas a la vez.

> **Note**
>
> Después de completar esta operación, ya no tendrá acceso a las versiones de la tabla y a las particiones que pertenecen a la tabla. AWS Glue elimina estos recursos "huérfanos" de manera asíncrona en forma oportuna, a discreción del servicio.
> Para asegurarse de la eliminación inmediata de todos los recursos relacionados, antes de llamar a `BatchDeleteTable`, use `DeleteTableVersion` o `BatchDeleteTableVersion` y `DeletePartition` o `BatchDeletePartition`, para eliminar todos los recursos que pertenezcan a la tabla.

Solicitud

- `CatalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID del Catálogo de datos donde reside la tabla. Si no se proporciona ninguno, se utiliza el ID de AWS cuenta de forma predeterminada.

- `DatabaseName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la base de datos de catálogos donde residen las tablas que se van a eliminar. Para compatibilidad con Hive, este nombre está completamente en minúsculas.

- `TablesToDelete` – Obligatorio: matriz de cadenas UTF-8, con 100 cadenas como máximo.

  Una lista de las tablas que se van a eliminar.

- `TransactionId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Custom string pattern #16.

  ID de transacción en el que se va a eliminar el contenido de la tabla.

Respuesta

- `Errors`: matriz de objetos TableError.

  Una lista de errores encontrados al intentar eliminar las tablas especificadas.

Errores

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`
- `ResourceNotReadyException`

GetTable acción (Python: get_table)

Recupera la definición de `Table` en un Catálogo de datos para una tabla especificada.

Solicitud

- `CatalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID del Catálogo de datos donde reside la tabla. Si no se proporciona ninguno, se utiliza el ID de la AWS cuenta de forma predeterminada.
- `DatabaseName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la base de datos en el catálogo donde reside la tabla. Para compatibilidad con Hive, este nombre está completamente en minúsculas.
- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la tabla para la que recuperar la definición. Para compatibilidad con Hive, este nombre está completamente en minúsculas.
- `TransactionId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Custom string pattern #16.

  ID de transacción en el que se va a leer el contenido de la tabla.
- `QueryAsOfTime`: marca temporal.

  El momento a partir del que se debe leer el contenido de la tabla. Si no se establece, se utilizará el tiempo de confirmación de la transacción más reciente. No se puede especificar junto con `TransactionId`.

Respuesta

- `Table`: un objeto Tabla.

  Objeto `Table` que define la tabla especificada.

Errores

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`
- `ResourceNotReadyException`
- `FederationSourceException`
- `FederationSourceRetryableException`

GetTables acción (Python: get_tables)

Recupera las definiciones de algunas o de todas las tablas en una `Database` determinada.

Solicitud

- `CatalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID del Catálogo de datos donde residen las tablas. Si no se proporciona ninguno, se utiliza el ID de la AWS cuenta de forma predeterminada.
- `DatabaseName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  La base de datos del catálogo cuyas tablas se van a listar. Para compatibilidad con Hive, este nombre está completamente en minúsculas.
- `Expression`: cadena UTF-8 con un máximo de 2048 bytes de largo, que coincide con el Single-line string pattern.

  Un patrón de expresiones regulares. Si están presentes, solo se devuelven esas tablas cuyos nombres coinciden con el patrón.
- `NextToken`: cadena UTF-8.

  Token de continuación, incluido si se trata de una llamada de continuidad.
- `MaxResults`: número (entero) que no es inferior a 1 ni es superior a 100.

El número máximo de tablas que se devuelven en una única respuesta.

- `TransactionId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Custom string pattern #16.

  ID de transacción en el que se va a leer el contenido de la tabla.

- `QueryAsOfTime`: marca temporal.

  El momento a partir del que se debe leer el contenido de la tabla. Si no se establece, se utilizará el tiempo de confirmación de la transacción más reciente. No se puede especificar junto con `TransactionId`.

Respuesta

- `TableList`: matriz de objetos Tabla.

  Una lista de los objetos `Table` solicitados.

- `NextToken`: cadena UTF-8.

  Un token de continuación, presente si el segmento de lista actual no es el último.

Errores

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`
- `GlueEncryptionException`
- `FederationSourceException`
- `FederationSourceRetryableException`

GetTableVersion acción (Python: get_table_version)

Recupera una versión especificada de una tabla.

Solicitud

- `CatalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID del Catálogo de datos donde residen las tablas. Si no se proporciona ninguno, se utiliza el ID de la AWS cuenta de forma predeterminada.

- `DatabaseName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  La base de datos del catálogo donde reside la tabla. Para compatibilidad con Hive, este nombre está completamente en minúsculas.

- `TableName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la tabla. Para compatibilidad con Hive, este nombre está completamente en minúsculas.

- `VersionId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El valor de ID de la versión de la tabla que se va a recuperar. Una `VersionID` es una representación de cadena de un número entero. Cada versión se incrementa en 1.

Respuesta

- `TableVersion`: un objeto TableVersion.

  La versión de la tabla solicitada.

Errores

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`

GetTableVersions acción (Python: get_table_versions)

Recupera una lista de cadenas que identifican las versiones disponibles de una tabla especificada.

Solicitud

- `CatalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID del Catálogo de datos donde residen las tablas. Si no se proporciona ninguno, se utiliza el ID de la AWS cuenta de forma predeterminada.
- `DatabaseName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  La base de datos del catálogo donde reside la tabla. Para compatibilidad con Hive, este nombre está completamente en minúsculas.
- `TableName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la tabla. Para compatibilidad con Hive, este nombre está completamente en minúsculas.
- `NextToken`: cadena UTF-8.

  Token de continuación si no se trata de la primera llamada.
- `MaxResults`: número (entero) que no es inferior a 1 ni es superior a 100.

  El número máximo de versiones de la tabla que se devuelven en una respuesta.

Respuesta

- `TableVersions`: matriz de objetos TableVersion.

  Una lista de cadenas que identifican las versiones disponibles de la tabla especificada.
- `NextToken`: cadena UTF-8.

  Token de continuación, si la lista de versiones disponibles no incluye la última.

Errores

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`

DeleteTableVersion acción (Python: delete_table_version)

Elimina una versión especificada de una tabla.

Solicitud

- `CatalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID del Catálogo de datos donde residen las tablas. Si no se proporciona ninguno, se utiliza el ID de AWS cuenta de forma predeterminada.
- `DatabaseName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  La base de datos del catálogo donde reside la tabla. Para compatibilidad con Hive, este nombre está completamente en minúsculas.
- `TableName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la tabla. Para compatibilidad con Hive, este nombre está completamente en minúsculas.
- `VersionId`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID de la versión de la tabla que se va a eliminar. Una `VersionID` es una representación de cadena de un número entero. Cada versión se incrementa en 1.

Respuesta

- Sin parámetros de respuesta.

Errores

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

BatchDeleteTableVersion acción (Python: batch_delete_table_version)

Elimina un lote especificado de versiones de una tabla.

Solicitud

- `CatalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID del Catálogo de datos donde residen las tablas. Si no se proporciona ninguno, se utiliza el ID de cuenta de forma predeterminada. AWS
- `DatabaseName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  La base de datos del catálogo donde reside la tabla. Para compatibilidad con Hive, este nombre está completamente en minúsculas.
- `TableName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la tabla. Para compatibilidad con Hive, este nombre está completamente en minúsculas.
- `VersionIds` – Obligatorio: matriz de cadenas UTF-8, con 100 cadenas como máximo.

  Una lista de los ID de versiones que se van a eliminar. Una `VersionId` es una representación de cadena de un número entero. Cada versión se incrementa en 1.

Respuesta

- `Errors`: matriz de objetos TableVersionError.

  Una lista de errores encontrados mientras se intentan eliminar las versiones de la tabla especificadas.

Errores

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

SearchTables acción (Python: search_tables)

Busca un conjunto de tablas basado en las propiedades de los metadatos de la tabla, así como en la base de datos principal. Puede realizar búsquedas en condiciones de texto o filtro.

Solo puede obtener tablas a las que tiene acceso en función de las políticas de seguridad definidas en Lake Formation. Necesita al menos un acceso de solo lectura a la tabla para que se devuelva. Si no tiene acceso a todas las columnas de la tabla, estas columnas no se buscarán cuando le devuelva la lista de tablas. Si tiene acceso a las columnas, pero no a los datos de las columnas, esas columnas y los metadatos asociados para esas columnas se incluirán en la búsqueda.

Solicitud

- `CatalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Un identificador único, que consta de *`account_id`*.
- `NextToken`: cadena UTF-8.

  Token de continuación, incluido si se trata de una llamada de continuidad.
- `Filters`: matriz de objetos PropertyPredicate.

  Una lista de pares clave-valor y un comparador utilizado para filtrar los resultados de búsqueda. Devuelve todas las entidades que coinciden con el predicado.

El miembro `Comparator` de la estructura `PropertyPredicate` se usa solo para campos de tiempo y se puede omitir para otros tipos de campos. Además, al comparar valores de cadena, como `Key=Name`, se utiliza un algoritmo de coincidencia aproximada. El campo `Key` (por ejemplo, el valor del campo `Name`) se divide en tokens con determinados caracteres de puntuación, por ejemplo, -, :, #, etc. Luego, cada token es una coincidencia exacta en comparación con el miembro `Value` de `PropertyPredicate`. Por ejemplo, para `Key=Name` y `Value=link`, se devuelven las tablas denominadas `customer-link` y `xx-link-yy`, pero no se devuelve `xxlinkyy`.

- `SearchText`: cadena de valor, de 1024 bytes de largo como máximo.

  Una cadena utilizada para una búsqueda de texto.

  Especificar un valor entre comillas filtra en base a una coincidencia exacta con el valor.

- `SortCriteria`: matriz de objetos SortCriterion, con 1 estructura como máximo.

  Una lista de criterios para ordenar los resultados por nombre de campo, en orden ascendente o descendente.

- `MaxResults`: número (entero) que no es inferior a 1 ni es superior a 1000.

  El número máximo de tablas que se devuelven en una única respuesta.

- `ResourceShareType`: cadena UTF-8 (valores válidos: `FOREIGN` | `ALL` | `FEDERATED`).

  Le permite especificar que desea buscar en las tablas compartidas con su cuenta. Los valores permitidos son `FOREIGN` o `ALL`.

  - Si se establece en `FOREIGN`, buscará en las tablas compartidas con su cuenta.
  - Si se establece en `ALL`, buscará en las tablas compartidas con su cuenta, así como en las tablas de su cuenta local.

Respuesta

- `NextToken`: cadena UTF-8.

  Un token de continuación, presente si el segmento de lista actual no es el último.

- `TableList`: matriz de objetos Tabla.

  Una lista de los objetos `Table` solicitados. La respuesta `SearchTables` devuelve solo las tablas a las que tiene acceso.

Errores

- `InternalServiceException`
- `InvalidInputException`
- `OperationTimeoutException`

GetPartitionIndexes acción (Python: get_partition_indexes)

Recupera los índices de partición asociados a una tabla.

Solicitud

- `CatalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID del catálogo donde reside la tabla.
- `DatabaseName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Especifica el nombre de una base de datos desde la que desea recuperar índices de particiones.
- `TableName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Especifica el nombre de una base de datos para la que desea recuperar índices de particiones.
- `NextToken`: cadena UTF-8.

  Token de continuación, incluido si se trata de una llamada de continuidad.

Respuesta

- `PartitionIndexDescriptorList`: matriz de objetos PartitionIndexDescriptor.

  Una lista de descriptores de índice.
- `NextToken`: cadena UTF-8.

  Un token de continuación, presente si el segmento de lista actual no es el último.

Errores

- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`
- `EntityNotFoundException`
- `ConflictException`

CreatePartitionIndex acción (Python: create_partition_index)

Crea un índice de partición especificado en una tabla existente.

Solicitud

- `CatalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID del catálogo donde reside la tabla.
- `DatabaseName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Especifica el nombre de una base de datos en la que desea crear un índice de particiones.
- `TableName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Especifica el nombre de una tabla en la que desea crear un índice de particiones.
- `PartitionIndex`: obligatorio: objeto PartitionIndex.

  Especifica una estructura de `PartitionIndex` para crear un índice de partición en una tabla existente.

Respuesta

- Sin parámetros de respuesta.

Errores

- `AlreadyExistsException`
- `InvalidInputException`
- `EntityNotFoundException`
- `ResourceNumberLimitExceededException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`

DeletePartitionIndex acción (Python: delete_partition_index)

Elimina un índice de partición especificado de una tabla existente.

Solicitud

- `CatalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID del catálogo donde reside la tabla.
- `DatabaseName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Especifica el nombre de una base de datos desde la que desea eliminar un índice de particiones.
- `TableName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Especifica el nombre de una tabla desde la que desea eliminar un índice de particiones.
- `IndexName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre del índice de partición que se eliminará.

Respuesta

- Sin parámetros de respuesta.

Errores

- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`
- `EntityNotFoundException`
- `ConflictException`
- `GlueEncryptionException`

GetColumnStatisticsForTable acción (Python: get_column_statistics_for_table)

Recupera estadísticas de las columnas de las tablas.

El permiso de Identity and Access Management (IAM) necesario para esta operación es `GetTable`.

Solicitud

- `CatalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID del Catálogo de datos donde residen las particiones en cuestión. Si no se proporciona ninguno, se utiliza el ID de cuenta de forma predeterminada. AWS
- `DatabaseName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la base de datos de catálogos donde residen las particiones.
- `TableName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre de la tabla de las particiones.
- `ColumnNames` – Obligatorio: matriz de cadenas UTF-8, con 100 cadenas como máximo.

  Una lista de los nombres de las columnas.

Respuesta

- `ColumnStatisticsList`: matriz de objetos ColumnStatistics.

Lista de ColumnStatistics.

- `Errors`: matriz de objetos ColumnError.

  No ColumnStatistics se pudo recuperar la lista.

Errores

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`

UpdateColumnStatisticsForTable acción (Python: update_column_statistics_for_table)

Crea o actualiza las estadísticas de las columnas de la tabla.

El permiso de Identity and Access Management (IAM) necesario para esta operación es `UpdateTable`.

Solicitud

- `CatalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID del Catálogo de datos donde residen las particiones en cuestión. Si no se proporciona ninguno, se utiliza el ID de cuenta de forma predeterminada. AWS
- `DatabaseName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la base de datos de catálogos donde residen las particiones.
- `TableName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre de la tabla de las particiones.
- `ColumnStatisticsList` – Obligatorio: matriz de objetos ColumnStatistics, con 25 estructuras como máximo.

Una lista de las estadísticas de las columnas.

Respuesta

- `Errors`: matriz de objetos ColumnStatisticsError.

  Lista de ColumnStatisticsErrors.

Errores

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`

DeleteColumnStatisticsForTable acción (Python: delete_column_statistics_for_table)

Recupera estadísticas de las columnas de las tablas.

El permiso de Identity and Access Management (IAM) necesario para esta operación es `DeleteTable`.

Solicitud

- `CatalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID del Catálogo de datos donde residen las particiones en cuestión. Si no se proporciona ninguno, se utiliza el ID de cuenta de forma predeterminada. AWS
- `DatabaseName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la base de datos de catálogos donde residen las particiones.
- `TableName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

Nombre de la tabla de las particiones.

- `ColumnName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la columna.

Respuesta

- Sin parámetros de respuesta.

Errores

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`

API de partición

La API de partición describe los tipos de datos y las operaciones que se utilizan para trabajar con particiones.

Tipos de datos

- Estructura Partition
- Estructura PartitionInput
- Estructura PartitionSpecWithSharedStorageDescriptor
- Estructura PartitionListComposingSpec
- Estructura PartitionSpecProxy
- Estructura PartitionValueList
- Estructura Segment
- Estructura PartitionError
- Estructura BatchUpdatePartitionFailureEntry

- Estructura BatchUpdatePartitionRequestEntry
- Estructura StorageDescriptor
- Estructura SchemaReference
- Estructura SerDeInfo
- Estructura SkewedInfo

Estructura Partition

Representa un sector de los datos de la tabla.

Campos

- `Values`: matriz de cadenas UTF-8.

  Los valores de la partición.
- `DatabaseName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la base de datos de catálogos donde se crea la partición.
- `TableName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la tabla de base de datos en la que se desea crear la partición.
- `CreationTime`: marca temporal.

  La hora a la que se creó la partición.
- `LastAccessTime`: marca temporal.

  La última vez que se obtuvo acceso a la partición.
- `StorageDescriptor`: un objeto StorageDescriptor.

  Ofrece información sobre la ubicación física donde se almacena la partición.
- `Parameters`: matriz de mapas de pares clave-valor.

  Cada clave es una cadena de claves con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Cada valor es una cadena UTF-8 que no tiene más de 512000 bytes de largo.

Estos pares de clave-valor definen los parámetros de partición.

- `LastAnalyzedTime`: marca temporal.

  La última vez que se calcularon las estadísticas de columna para esta partición.

- `CatalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  ID del Data Catalog donde reside la partición.

Estructura PartitionInput

La estructura que se utiliza para crear y actualizar una partición.

Campos

- `Values`: matriz de cadenas UTF-8.

  Los valores de la partición. Aunque este parámetro no lo requiere el SDK, debe especificarlo para una entrada válida.

  Los valores de las claves de la nueva partición deben pasarse como una matriz de objetos String que deben ordenarse en el mismo orden que las claves de partición que aparecen en el prefijo de Amazon S3. De lo contrario, AWS Glue agregará los valores a las claves erróneas.

- `LastAccessTime`: marca temporal.

  La última vez que se obtuvo acceso a la partición.

- `StorageDescriptor`: un objeto StorageDescriptor.

  Ofrece información sobre la ubicación física donde se almacena la partición.

- `Parameters`: matriz de mapas de pares clave-valor.

  Cada clave es una cadena de claves con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Cada valor es una cadena UTF-8 que no tiene más de 512000 bytes de largo.

  Estos pares de clave-valor definen los parámetros de partición.

- `LastAnalyzedTime`: marca temporal.

La última vez que se calcularon las estadísticas de columna para esta partición.

Estructura PartitionSpecWithSharedStorageDescriptor

Especificación de partición para las particiones que comparten una ubicación física.

Campos

- `StorageDescriptor`: un objeto StorageDescriptor.

  La información compartida sobre el almacenamiento físico.
- `Partitions`: matriz de objetos Partición.

  Lista de las particiones que comparten esta ubicación física.

Estructura PartitionListComposingSpec

Muestra las particiones relacionadas.

Campos

- `Partitions`: matriz de objetos Partición.

  Lista de las particiones en la especificación de composición.

Estructura PartitionSpecProxy

Ofrece una ruta hacia las particiones especificadas.

Campos

- `DatabaseName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  La base de datos de catálogos donde residen las particiones.
- `TableName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la tabla que contiene las particiones.

- `RootPath`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  La ruta raíz del proxy para abordar las particiones.
- `PartitionSpecWithSharedSD`: un objeto PartitionSpecWithSharedStorageDescriptor.

  Especificación de las particiones que comparten la misma ubicación de almacenamiento física.
- `PartitionListComposingSpec`: un objeto PartitionListComposingSpec.

  Especifica una lista de particiones.

Estructura PartitionValueList

Contiene una lista de valores que definen particiones.

Campos

- `Values` – Obligatorio: una matriz de cadenas UTF-8.

  La lista de valores.

Estructura Segment

Define una región de particiones de una tabla que no se solapa, lo que permite ejecutar varias solicitudes a la vez.

Campos

- `SegmentNumber` – Obligatorio: número (entero), cero como máximo.

  El número de índice de base cero del segmento. Por ejemplo, si el número total de segmentos es de 4, los valores `SegmentNumber` van de 0 a 3.
- `TotalSegments` – Obligatorio: número (entero), uno como mínimo o 10 como máximo.

  El número total de segmentos.

Estructura PartitionError

Contiene información sobre un error de partición.

Campos

- `PartitionValues`: matriz de cadenas UTF-8.

  Los valores que definen la partición.
- `ErrorDetail`: un objeto ErrorDetail.

  Detalles sobre el error de la partición.

Estructura BatchUpdatePartitionFailureEntry

Contiene información sobre un error de partición de actualización por lotes.

Campos

- `PartitionValueList`: matriz de cadenas UTF-8, con 100 cadenas como máximo.

  Una lista de los valores que definen las particiones.
- `ErrorDetail`: un objeto ErrorDetail.

  Los detalles sobre el error de partición del actualización por lotes.

Estructura BatchUpdatePartitionRequestEntry

Una estructura que contiene los valores y la estructura utilizados para actualizar una partición.

Campos

- `PartitionValueList` – Obligatorio: matriz de cadenas UTF-8, con 100 cadenas como máximo.

  Una lista de los valores que definen las particiones.
- `PartitionInput`: obligatorio: objeto PartitionInput.

  La estructura que se utiliza para actualizar una partición.

Estructura StorageDescriptor

Describe el almacenamiento físico de los datos de la tabla.

Campos

- `Columns`: matriz de objetos Columna.

  Una lista de las `Columns` de la tabla.

- `Location`: cadena de ubicación de un máximo de 2056 bytes de largo, que coincide con el URI address multi-line string pattern.

  Ubicación física de la tabla. De forma predeterminada, adopta la forma de la ubicación de almacén, seguida de la ubicación de la base de datos en el almacén, seguida del nombre de tabla.

- `AdditionalLocations`: matriz de cadenas UTF-8.

  Lista de ubicaciones que apuntan a la ruta en la que se encuentra una tabla Delta.

- `InputFormat`: cadena de formato de un máximo de 128 bytes de largo, que coincide con el Single-line string pattern.

  Formato de entrada: `SequenceFileInputFormat` (binario), `TextInputFormat` o formato personalizado.

- `OutputFormat`: cadena de formato de un máximo de 128 bytes de largo, que coincide con el Single-line string pattern.

  Formato de salida: `SequenceFileOutputFormat` (binario), `IgnoreKeyTextOutputFormat` o formato personalizado.

- `Compressed`: booleano.

  `True` si los datos de la tabla están comprimidos o `False` si no.

- `NumberOfBuckets`: número (entero).

  Debe especificarse si la tabla contiene alguna columna de dimensión.

- `SerdeInfo`: un objeto SerDeInfo.

  La información de serialización y deserialización (SerDe).

- `BucketColumns`: matriz de cadenas UTF-8.

  Lista de columnas de agrupamiento del reductor, columnas de clústeres y columnas de almacenamiento en bucket de la tabla.

- `SortColumns`: matriz de objetos Order.

Una lista donde se especifica el orden de clasificación de cada bucket en la tabla.

- `Parameters`: matriz de mapas de pares clave-valor.

  Cada clave es una cadena de claves con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Cada valor es una cadena UTF-8 que no tiene más de 512000 bytes de largo.

  Las propiedades facilitadas por el usuario en formato clave-valor.
- `SkewedInfo`: un objeto SkewedInfo.

  La información sobre los valores que aparecen con frecuencia en una columna (valores sesgados).
- `StoredAsSubDirectories`: booleano.

  `True` si los datos de la tabla se almacenan en subdirectorios o `False` si no.
- `SchemaReference`: un objeto SchemaReference.

  Objeto que hace referencia a un esquema almacenado en AWS Glue Schema Registry.

  Al crear una tabla, puede pasar una lista vacía de columnas para el esquema y, en su lugar, utilizar una referencia de esquema.

Estructura SchemaReference

Objeto que hace referencia a un esquema almacenado en AWS Glue Schema Registry.

Campos

- `SchemaId`: un objeto SchemaId.

  Estructura que contiene campos de identidad de esquema. Este o el `SchemaVersionId` tiene que ser proporcionado.
- `SchemaVersionId`: cadena UTF-8, con 36 bytes de largo como mínimo y 36 bytes de largo como máximo, que coincide con el Custom string pattern #17.

  El ID único asignado a una versión del esquema. Este o el `SchemaId` tiene que ser proporcionado.
- `SchemaVersionNumber`: número (largo) que no es inferior a 1 ni es superior a 100 000.

  El número de versión del esquema.

Estructura SerDeInfo

Información sobre un programa de serialización y deserialización (SerDe) que sirve de extractor y cargador.

Campos

- `Name`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre del SerDe.

- `SerializationLibrary`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  En general, la clase que implementa el SerDe. Un ejemplo es `org.apache.hadoop.hive.serde2.columnar.ColumnarSerDe`.

- `Parameters`: matriz de mapas de pares clave-valor.

  Cada clave es una cadena de claves con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Cada valor es una cadena UTF-8 que no tiene más de 512000 bytes de largo.

  Estos pares de clave-valor definen parámetros de inicialización para el SerDe.

Estructura SkewedInfo

Especifica valores sesgados en una tabla. Los valores sesgados son los que se producen con una frecuencia muy alta.

Campos

- `SkewedColumnNames`: matriz de cadenas UTF-8.

  Una lista de nombres de columnas que contienen valores de sesgado.

- `SkewedColumnValues`: matriz de cadenas UTF-8.

  Una lista de valores que aparecen con tanta frecuencia como para considerarse de sesgado.

- `SkewedColumnValueLocationMaps`: matriz de mapas de pares clave-valor.

Cada clave es una cadena UTF-8.

Cada valor es una cadena UTF-8.

Un mapeo de valores de sesgado a las columnas que los contienen.

Operaciones

- Acción CreatePartition (Python: create_partition)
- Acción BatchCreatePartition (Python: batch_create_partition)
- Acción UpdatePartition (Python: update_partition)
- Acción DeletePartition (Python: delete_partition)
- Acción BatchDeletePartition (Python: batch_delete_partition)
- Acción GetPartition (Python: get_partition)
- Acción GetPartitions (Python: get_partitions)
- Acción BatchGetPartition (Python: batch_get_partition)
- Acción BatchUpdatePartition (Python: batch_update_partition)
- Acción GetColumnStatisticsForPartition (Python: get_column_statistics_for_partition)
- Acción UpdateColumnStatisticsForPartition (Python: update_column_statistics_for_partition)
- Acción DeleteColumnStatisticsForPartition (Python: delete_column_statistics_for_partition)

Acción CreatePartition (Python: create_partition)

Crea una partición nueva.

Solicitud

- `CatalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID de cuenta de AWS del catálogo en el que se creará la partición.
- `DatabaseName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la base de datos de metadatos donde se va a crear la partición.

- `TableName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la tabla de datos de metadatos donde se va a crear la partición.
- `PartitionInput`: obligatorio: objeto PartitionInput.

  Estructura `PartitionInput` que define la partición que se va a crear.

Respuesta

- Sin parámetros de respuesta.

Errores

- `InvalidInputException`
- `AlreadyExistsException`
- `ResourceNumberLimitExceededException`
- `InternalServiceException`
- `EntityNotFoundException`
- `OperationTimeoutException`
- `GlueEncryptionException`

Acción BatchCreatePartition (Python: batch_create_partition)

Crea una o varias particiones en una operación por lotes.

Solicitud

- `CatalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID del catálogo en el que se va a crear la partición. En la actualidad, debe ser el ID de cuenta de AWS.
- `DatabaseName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la base de datos de metadatos donde se va a crear la partición.

- `TableName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la tabla de datos de metadatos donde se va a crear la partición.

- `PartitionInputList` – Obligatorio: matriz de objetos PartitionInput, con 100 estructuras como máximo.

  Lista de estructuras `PartitionInput` que definen las particiones que se van a crear.

Respuesta

- `Errors`: matriz de objetos PartitionError.

  Los errores encontrados al intentar crear las particiones solicitadas.

Errores

- `InvalidInputException`
- `AlreadyExistsException`
- `ResourceNumberLimitExceededException`
- `InternalServiceException`
- `EntityNotFoundException`
- `OperationTimeoutException`
- `GlueEncryptionException`

Acción UpdatePartition (Python: update_partition)

Actualiza una partición.

Solicitud

- `CatalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID del catálogo de datos donde reside la partición que se va a actualizar. Si no se proporciona ninguno, se usará de forma predeterminada el ID de cuenta de AWS.

- `DatabaseName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la base de datos de catálogos donde reside la tabla relevante.

- `TableName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la tabla donde se encuentra la partición que se va a actualizar.

- `PartitionValueList` – Obligatorio: matriz de cadenas UTF-8, con 100 cadenas como máximo.

  Lista de valores de clave de partición que definen la partición que se va a actualizar.

- `PartitionInput`: obligatorio: objeto PartitionInput.

  El nuevo objeto de la partición para el que se actualiza la partición.

  La propiedad `Values` no puede modificarse. Si desea modificar los valores de clave de partición de una partición, elimine y vuelva a crear la partición.

Respuesta

- Sin parámetros de respuesta.

Errores

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`

Acción DeletePartition (Python: delete_partition)

Elimina la partición especificada.

Solicitud

- `CatalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID del catálogo de datos donde reside la partición que se va a eliminar. Si no se proporciona ninguno, se usará de forma predeterminada el ID de cuenta de AWS.
- `DatabaseName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la base de datos de catálogos donde reside la tabla relevante.
- `TableName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la tabla que contiene la partición que se va a eliminar.
- `PartitionValues` – Obligatorio: una matriz de cadenas UTF-8.

  Los valores que definen la partición.

Respuesta

- Sin parámetros de respuesta.

Errores

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

Acción BatchDeletePartition (Python: batch_delete_partition)

Elimina una o varias particiones en una operación por lotes.

Solicitud

- `CatalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

El ID del catálogo de datos donde reside la partición que se va a eliminar. Si no se proporciona ninguno, se usará de forma predeterminada el ID de cuenta de AWS.

- `DatabaseName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la base de datos de catálogos donde reside la tabla relevante.

- `TableName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la tabla que contiene las particiones que se van a eliminar.

- `PartitionsToDelete` – Obligatorio: matriz de objetos PartitionValueList, con 25 estructuras como máximo.

  Lista de estructuras `PartitionInput` que definen las particiones que se van a eliminar.

Respuesta

- `Errors`: matriz de objetos PartitionError.

  Los errores encontrados al intentar eliminar las particiones solicitadas.

Errores

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`

Acción GetPartition (Python: get_partition)

Recupera información sobre una partición especificada.

Solicitud

- `CatalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

El ID del catálogo de datos donde reside la partición en cuestión. Si no se proporciona ninguno, se usará de forma predeterminada el ID de cuenta de AWS.

- `DatabaseName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la base de datos de catálogos donde reside la partición.

- `TableName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre de la tabla de particiones.

- `PartitionValues` – Obligatorio: una matriz de cadenas UTF-8.

  Los valores que definen la partición.

Respuesta

- `Partition`: un objeto Partición.

  La información solicitada, con formato de objeto `Partition`.

Errores

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`
- `FederationSourceException`
- `FederationSourceRetryableException`

Acción GetPartitions (Python: get_partitions)

Recupera información acerca de las particiones de una tabla.

Solicitud

- `CatalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID del Catálogo de datos donde residen las particiones en cuestión. Si no se proporciona ninguno, se usará de forma predeterminada el ID de cuenta de AWS.
- `DatabaseName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la base de datos de catálogos donde residen las particiones.
- `TableName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre de la tabla de las particiones.
- `Expression`: cadena de predicado de un máximo de 2048 bytes de largo, que coincide con el URI address multi-line string pattern.

  Una expresión que filtra las particiones que se van a devolver.

  La expresión usa una sintaxis SQL similar a la cláusula de filtro `WHERE` de SQL. El analizador de instrucciones SQL JSQLParser analiza la expresión.

  Operadores: estos son los operadores que puede usar en la llamada a la API `Expression`:

  =

  Comprueba si los valores de los dos operandos son iguales o no; en caso afirmativo, la condición será "true".

  Ejemplo: supongamos que la "variable a" tiene 10 y, la "variable b", 20.

  (a = b) no es "true".

  < >

  Comprueba si los valores de los dos operandos son iguales o no; en caso negativo, la condición será "true".

  Ejemplo: (a < > b) es "true".

>

Comprueba si el valor del operando izquierdo es mayor que el valor del operando derecho; en caso afirmativo, la condición será "true".

Ejemplo: (a **>** b) no es "true".

<

Comprueba si el valor del operando izquierdo es menor que el valor del operando derecho; en caso afirmativo, la condición será "true".

Ejemplo: (a **<** b) es "true".

>=

Comprueba si el valor del operando izquierdo es mayor o igual que el valor del operando derecho; en caso afirmativo, la condición será "true".

Ejemplo: (a **>=** b) no es "true".

<=

Comprueba si el valor del operando izquierdo es menor o igual que el valor del operando derecho; en caso afirmativo, la condición será "true".

Ejemplo: (a **<=** b) es "true".

AND, OR, IN, BETWEEN, LIKE, NOT, IS NULL

Operadores lógicos.

Tipos de clave de partición admitidos: estas son las claves de partición admitidas.

- `string`
- `date`
- `timestamp`
- `int`
- `bigint`
- `long`
- `tinyint`
- `smallint`

- `decimal`

Si se encuentra un tipo no válido, se produce una excepción.

En la siguiente lista se muestran todos los tipos de operador válido. Al definir un rastreador, se crea el tipo `partitionKey` como `STRING`, para ser compatible con las particiones del catálogo.

Ejemplo de llamada a la API:

Example

La tabla `twitter_partition` tiene tres particiones:

```
year = 2015
        year = 2016
        year = 2017
```

Example

Get Partition `year` equivale a 2015

```
aws glue get-partitions --database-name dbname --table-name twitter_partition
        --expression "year*=*'2015'"
```

Example

Get Partition `year` entre 2016 y 2018 (exclusivo)

```
aws glue get-partitions --database-name dbname --table-name twitter_partition
        --expression "year>'2016' AND year<'2018'"
```

Example

Get Partition `year` entre 2015 y 2018 (incluido). Las siguientes llamadas a la API son equivalentes entre sí:

```
aws glue get-partitions --database-name dbname --table-name twitter_partition
        --expression "year>='2015' AND year<='2018'"
```

```
        aws glue get-partitions --database-name dbname --table-name
twitter_partition
        --expression "year BETWEEN 2015 AND 2018"

        aws glue get-partitions --database-name dbname --table-name
twitter_partition
        --expression "year IN (2015,2016,2017,2018)"
```

Example

Filtro comodín de partición, donde la siguiente salida de llamadas será el año de partición, 2017. No se admite una expresión regular en `LIKE`.

```
aws glue get-partitions --database-name dbname --table-name twitter_partition
        --expression "year LIKE '%7'"
```

- `NextToken`: cadena UTF-8.

  Token de continuación, si no es la primera llamada para recuperar estas particiones.

- `Segment`: un objeto Segmento.

  El segmento de las particiones de la tabla que se va a analizar en esta solicitud.

- `MaxResults`: número (entero) que no es inferior a 1 ni es superior a 1000.

  El número máximo de particiones que se devuelven en una única respuesta.

- `ExcludeColumnSchema`: booleano.

  Si es verdadero, especifica no devolver el esquema de columna de partición. Es útil cuando solo está interesado en otros atributos de partición, como valores de partición o ubicación. Este enfoque evita el problema de una respuesta grande al no devolver datos duplicados.

- `TransactionId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Custom string pattern #16.

  ID de transacción en el que se va a leer el contenido de la partición.

- `QueryAsOfTime`: marca temporal.

  El momento a partir del que se debe leer el contenido de la partición. Si no se establece, se utilizará el tiempo de confirmación de la transacción más reciente. No se puede especificar junto con `TransactionId`.

Respuesta

- `Partitions`: matriz de objetos Partición.

  Lista de particiones solicitadas.

- `NextToken`: cadena UTF-8.

  Un token de continuación, si la lista de particiones que se devuelve no incluye la última.

Errores

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`
- `GlueEncryptionException`
- `InvalidStateException`
- `ResourceNotReadyException`
- `FederationSourceException`
- `FederationSourceRetryableException`

Acción BatchGetPartition (Python: batch_get_partition)

Recupera particiones en una solicitud por lotes.

Solicitud

- `CatalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID del Catálogo de datos donde residen las particiones en cuestión. Si no se proporciona ninguno, el ID de cuenta de AWS se usará de forma predeterminada.

- `DatabaseName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la base de datos de catálogos donde residen las particiones.

- `TableName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre de la tabla de las particiones.
- `PartitionsToGet` – Obligatorio: matriz de objetos PartitionValueList, con 1000 estructuras como máximo.

  Lista de los valores de partición que identifican las particiones que se van a recuperar.

Respuesta

- `Partitions`: matriz de objetos Partición.

  Lista de las particiones solicitadas.
- `UnprocessedKeys`: matriz de objetos PartitionValueList, con 1000 estructuras como máximo.

  Una lista de los valores de partición en la solicitud para la que no se devolvieron particiones.

Errores

- `InvalidInputException`
- `EntityNotFoundException`
- `OperationTimeoutException`
- `InternalServiceException`
- `GlueEncryptionException`
- `InvalidStateException`
- `FederationSourceException`
- `FederationSourceRetryableException`

Acción BatchUpdatePartition (Python: batch_update_partition)

Actualiza una o varias particiones en una operación por lotes.

Solicitud

- `CatalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

El ID del catálogo en el que se va a actualizar la partición. En la actualidad, debe ser el ID de cuenta de AWS.

- `DatabaseName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la base de datos de metadatos donde se va a actualizar la partición.

- `TableName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la tabla de metadatos donde se va a actualizar la partición.

- `Entries` – Obligatorio: una matriz de objetos BatchUpdatePartitionRequestEntry, con 1 estructura como mínimo y 100 estructuras como máximo.

  Lista de hasta 100 objetos `BatchUpdatePartitionRequestEntry` que se van a actualizar.

Respuesta

- `Errors`: matriz de objetos BatchUpdatePartitionFailureEntry.

  Los errores encontrados al intentar actualizar las particiones solicitadas. Una lista de objetos `BatchUpdatePartitionFailureEntry`.

Errores

- `InvalidInputException`
- `EntityNotFoundException`
- `OperationTimeoutException`
- `InternalServiceException`
- `GlueEncryptionException`

Acción GetColumnStatisticsForPartition (Python: get_column_statistics_for_partition)

Recupera estadísticas de las columnas de las particiones.

El permiso de Identity and Access Management (IAM) necesario para esta operación es `GetPartition`.

Solicitud

- `CatalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID del Catálogo de datos donde residen las particiones en cuestión. Si no se proporciona ninguno, el ID de cuenta de AWS se usará de forma predeterminada.
- `DatabaseName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la base de datos de catálogos donde residen las particiones.
- `TableName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre de la tabla de las particiones.
- `PartitionValues` – Obligatorio: una matriz de cadenas UTF-8.

  Una lista de los valores de las particiones que identifican la partición.
- `ColumnNames` – Obligatorio: matriz de cadenas UTF-8, con 100 cadenas como máximo.

  Una lista de los nombres de las columnas.

Respuesta

- `ColumnStatisticsList`: matriz de objetos ColumnStatistics.

  Lista de ColumnStatistics que no se pudieron recuperar.
- `Errors`: matriz de objetos ColumnError.

  Error al recuperar los datos estadísticos de la columna.

Errores

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

- `GlueEncryptionException`

Acción UpdateColumnStatisticsForPartition (Python: update_column_statistics_for_partition)

Crea o actualiza las estadísticas de las columnas de las particiones.

El permiso de Identity and Access Management (IAM) necesario para esta operación es `UpdatePartition`.

Solicitud

- `CatalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID del Catálogo de datos donde residen las particiones en cuestión. Si no se proporciona ninguno, el ID de cuenta de AWS se usará de forma predeterminada.
- `DatabaseName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la base de datos de catálogos donde residen las particiones.
- `TableName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre de la tabla de las particiones.
- `PartitionValues` – Obligatorio: una matriz de cadenas UTF-8.

  Una lista de los valores de las particiones que identifican la partición.
- `ColumnStatisticsList` – Obligatorio: matriz de objetos ColumnStatistics, con 25 estructuras como máximo.

  Una lista de las estadísticas de las columnas.

Respuesta

- `Errors`: matriz de objetos ColumnStatisticsError.

  Error al actualizar los datos de estadísticas de columna.

Errores

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`

Acción DeleteColumnStatisticsForPartition (Python: delete_column_statistics_for_partition)

Elimine las estadísticas de columnas de la partición de una columna.

El permiso de Identity and Access Management (IAM) necesario para esta operación es `DeletePartition`.

Solicitud

- `CatalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID del Catálogo de datos donde residen las particiones en cuestión. Si no se proporciona ninguno, el ID de cuenta de AWS se usará de forma predeterminada.
- `DatabaseName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la base de datos de catálogos donde residen las particiones.
- `TableName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre de la tabla de las particiones.
- `PartitionValues` – Obligatorio: una matriz de cadenas UTF-8.

  Una lista de los valores de las particiones que identifican la partición.
- `ColumnName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la columna.

Respuesta

- Sin parámetros de respuesta.

Errores

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`

API de conexión

La API de conexión describe los tipos de datos de conexión de AWS Glue y la API de creación, eliminación, actualización y creación de listados de conexiones.

Tipos de datos

- Estructura Connection
- Estructura ConnectionInput
- Estructura PhysicalConnectionRequirements
- Estructura GetConnectionsFilter

Estructura Connection

Define una conexión a un origen de datos.

Campos

- `Name`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la definición de la conexión.
- `Description`: cadena de descripción de un máximo de 2048 bytes de largo, que coincide con el URI address multi-line string pattern.

La descripción de la conexión.

- `ConnectionType`: cadena UTF-8 (valores válidos: `JDBC` | `SFTP` | `MONGODB` | `KAFKA` | `NETWORK` | `MARKETPLACE` | `CUSTOM` | `SALESFORCE`).

  Tipo de la conexión. Actualmente, SFTP no es compatible.

- `MatchCriteria`: matriz de cadenas UTF-8, con 10 cadenas como máximo.

  Una lista de criterios que se pueden usar para seleccionar esta conexión.

- `ConnectionProperties`: matriz de mapas de pares de clave-valor, con 100 pares como máximo.

  Cada clave es una cadena UTF-8 (valores válidos: `HOST` | `PORT` | `USERNAME="USER_NAME"` | `PASSWORD` | `ENCRYPTED_PASSWORD` | `JDBC_DRIVER_JAR_URI` | `JDBC_DRIVER_CLASS_NAME` | `JDBC_ENGINE` | `JDBC_ENGINE_VERSION` | `CONFIG_FILES` | `INSTANCE_ID` | `JDBC_CONNECTION_URL` | `JDBC_ENFORCE_SSL` | `CUSTOM_JDBC_CERT` | `SKIP_CUSTOM_JDBC_CERT_VALIDATION` | `CUSTOM_JDBC_CERT_STRING` | `CONNECTION_URL` | `KAFKA_BOOTSTRAP_SERVERS` | `KAFKA_SSL_ENABLED` | `KAFKA_CUSTOM_CERT` | `KAFKA_SKIP_CUSTOM_CERT_VALIDATION` | `KAFKA_CLIENT_KEYSTORE` | `KAFKA_CLIENT_KEYSTORE_PASSWORD` | `KAFKA_CLIENT_KEY_PASSWORD` | `ENCRYPTED_KAFKA_CLIENT_KEYSTORE_PASSWORD` | `ENCRYPTED_KAFKA_CLIENT_KEY_PASSWORD` | `SECRET_ID` | `CONNECTOR_URL` | `CONNECTOR_TYPE` | `CONNECTOR_CLASS_NAME` | `KAFKA_SASL_MECHANISM` | `KAFKA_SASL_PLAIN_USERNAME` | `KAFKA_SASL_PLAIN_PASSWORD` | `ENCRYPTED_KAFKA_SASL_PLAIN_PASSWORD` | `KAFKA_SASL_SCRAM_USERNAME` | `KAFKA_SASL_SCRAM_PASSWORD` | `KAFKA_SASL_SCRAM_SECRETS_ARN` | `ENCRYPTED_KAFKA_SASL_SCRAM_PASSWORD` | `KAFKA_SASL_GSSAPI_KEYTAB` | `KAFKA_SASL_GSSAPI_KRB5_CONF` | `KAFKA_SASL_GSSAPI_SERVICE` | `KAFKA_SASL_GSSAPI_PRINCIPAL` | `ROLE_ARN`).

  Cada valor es una cadena de valor que no tiene más de 1024 bytes de largo.

  Estos pares de clave-valor definen parámetros para la conexión:

  - `HOST`: el URI de host, ya sea el nombre de dominio completo (FQDN) o la dirección IPv4 del host de la base de datos.
  - `PORT`: el número de puerto, entre 1024 y 65535, del puerto en el que el host de la base de datos escucha las conexiones de la base de datos.

- USER_NAME: el nombre con el que iniciar sesión en la base de datos. La cadena de valor de USER_NAME es "USERNAME".
- PASSWORD: contraseña, si se utiliza una, del nombre de usuario.
- ENCRYPTED_PASSWORD: cuando habilita la protección de la contraseña de conexión al establecer ConnectionPasswordEncryption en la configuración de cifrado del Catálogo de datos, este campo almacena la contraseña cifrada.
- JDBC_DRIVER_JAR_URI: la ruta de Amazon Simple Storage Service (Amazon S3) al archivo JAR que contiene el controlador JDBC que se va a utilizar.
- JDBC_DRIVER_CLASS_NAME: el nombre de la clase de controlador de JDBC que se va a utilizar.
- JDBC_ENGINE: nombre del motor JDBC que se va a utilizar.
- JDBC_ENGINE_VERSION: la versión del motor JDBC que se va a utilizar.
- CONFIG_FILES: (reservado para uso futuro).
- INSTANCE_ID: la ID de instancia que se va a utilizar.
- JDBC_CONNECTION_URL: la URL para conectarse a un origen de datos JDBC.
- JDBC_ENFORCE_SSL: cadena booleana (true, false) que especifica si se aplicará la coincidencia de Capa de conexión segura (SSL) con el nombre de host para la conexión JDBC en el cliente. El valor predeterminado es false.
- CUSTOM_JDBC_CERT: una ubicación de Amazon S3 que especifica el certificado raíz del cliente. AWS Glue utiliza este certificado raíz para validar el certificado del cliente al conectarse a la base de datos del cliente. AWS Glue solo gestiona certificados X.509. El certificado proporcionado debe estar codificado en DER y suministrarse en formato PEM de codificación Base64.
- SKIP_CUSTOM_JDBC_CERT_VALIDATION: predeterminado en false. AWS Glue valida el algoritmo de firma y el algoritmo de clave pública de sujeto para el certificado de cliente. Los únicos algoritmos permitidos para el algoritmo de firma son SHA256withRSA, SHA384withRSA o SHA512withRSA. Para el algoritmo de clave pública del sujeto, la longitud de la clave debe ser de al menos 2048. Puede establecer el valor de esta propiedad en para true omitir la validación del certificado de cliente por parte de AWS Glue.
- CUSTOM_JDBC_CERT_STRING : cadena de certificado JDBC personalizada que se utiliza para la coincidencia de dominio o la coincidencia de nombre distinguido para evitar un ataque man-in-the-middle. En la base de datos de Oracle, se utiliza como SSL_SERVER_CERT_DN; en Microsoft SQL Server, se utiliza como hostNameInCertificate.

- `CONNECTION_URL`: la URL para conectarse a un origen de datos general (no JDBC).
- `SECRET_ID`: la ID secreta utilizada para el administrador secreto de credenciales.
- `CONNECTOR_URL`: la URL del conector para una conexión MARKETPLACE o CUSTOM.
- `CONNECTOR_TYPE`: el tipo de conector para una conexión MARKETPLACE o CUSTOM.
- `CONNECTOR_CLASS_NAME`: el nombre de la clase del conector para una conexión MARKETPLACE o CUSTOM.
- `KAFKA_BOOTSTRAP_SERVERS`: una lista separada por comas de los pares de hosts y puertos que son las direcciones de los agentes de Apache Kafka en un clúster de Kafka al que un cliente Kafka se conectará y se arrancará.
- `KAFKA_SSL_ENABLED`: para habilitar o desactivar SSL en una conexión de Apache Kafka. El valor predeterminado es “verdadero”.
- `KAFKA_CUSTOM_CERT`: la URL de Amazon S3 para el archivo de certificado de CA privado (formato .pem). El valor predeterminado es una cadena vacía.
- `KAFKA_SKIP_CUSTOM_CERT_VALIDATION`: para omitir la validación del archivo certificado de CA o no. AWS Glue valida para tres algoritmos: SHA256withRSA, SHA384withRSA y SHA512withRSA. El valor predeterminado es "falso".
- `KAFKA_CLIENT_KEYSTORE`: la ubicación de Amazon S3 del archivo de almacén de claves del cliente para la autenticación del lado del cliente Kafka (opcional).
- `KAFKA_CLIENT_KEYSTORE_PASSWORD`: la contraseña para acceder al almacén de claves proporcionado (opcional).
- `KAFKA_CLIENT_KEY_PASSWORD`: un almacén de claves puede consistir en varias claves, por lo que esta es la contraseña para acceder a la clave del cliente que se utilizará con la clave del lado del servidor Kafka (opcional).
- `ENCRYPTED_KAFKA_CLIENT_KEYSTORE_PASSWORD`: la versión cifrada de la contraseña del almacén de claves del cliente de Kafka (si el usuario ha seleccionado la configuración de cifrar contraseñas de AWS Glue).
- `ENCRYPTED_KAFKA_CLIENT_KEY_PASSWORD`: la versión cifrada de la contraseña de la clave del cliente de Kafka (si el usuario ha seleccionado la configuración de cifrar contraseñas de AWS Glue).
- `KAFKA_SASL_MECHANISM`: `"SCRAM-SHA-512"`, `"GSSAPI"`, `"AWS_MSK_IAM"` o `"PLAIN"`. Estos son los mecanismos SASL compatibles.
- `KAFKA_SASL_PLAIN_USERNAME`: un nombre de usuario sin formato utilizado para la autenticación con el mecanismo “PLAIN”.

- `KAFKA_SASL_PLAIN_PASSWORD`: una contraseña sin formato utilizada para la autenticación con el mecanismo “PLAIN”.
- `ENCRYPTED_KAFKA_SASL_PLAIN_PASSWORD`: la versión cifrada de la contraseña de SASL PLAIN de Kafka (si el usuario seleccionó la configuración de cifrado de contraseñas de AWS Glue).
- `KAFKA_SASL_SCRAM_USERNAME`: nombre de usuario de texto sin formato utilizado para autenticarse con el mecanismo “SCRAM-SHA-512”.
- `KAFKA_SASL_SCRAM_PASSWORD`: contraseña de texto sin formato utilizada para autenticarse con el mecanismo “SCRAM-SHA-512”.
- `ENCRYPTED_KAFKA_SASL_SCRAM_PASSWORD`: la versión cifrada de la contraseña SASL SCRAM de Kafka (si el usuario ha seleccionado la configuración de cifrado de contraseñas de AWS Glue).
- `KAFKA_SASL_SCRAM_SECRETS_ARN`: El nombre de recurso de Amazon (ARN) de un secreto en Secrets Manager AWS.
- `KAFKA_SASL_GSSAPI_KEYTAB`: la ubicación S3 de un archivo `keytab` de Kerberos. Una keytab almacena claves a largo plazo para uno o varias entidades principales. Para obtener más información, consulte MIT Kerberos Documentation: Keytab (Documentación de MIT Kerberos: Keytab).
- `KAFKA_SASL_GSSAPI_KRB5_CONF`: la ubicación S3 de un archivo `krb5.conf` de Kerberos. Un krb5.conf almacena información de configuración de Kerberos, como la ubicación del servidor KDC. Para obtener más información, consulte MIT Kerberos Documentation: krb5.conf (Documentación de MIT Kerberos: krb5.conf).
- `KAFKA_SASL_GSSAPI_SERVICE`: el nombre del servicio Kerberos, tal como se establece con `sasl.kerberos.service.name` en la configuración de Kafka.
- `KAFKA_SASL_GSSAPI_PRINCIPAL`: el nombre de la entidad principal de Kerberos que utiliza AWS Glue. Para obtener más información, consulte Kafka Documentation: Configuring Kafka Brokers (Documentación de Kafka: Configuración de los agentes de Kafka).

- `PhysicalConnectionRequirements`: un objeto PhysicalConnectionRequirements.

  Los requisitos de conexión física, como la nube privada virtual (VPC) y `SecurityGroup`, que se necesitan para crear la conexión correctamente.

- `CreationTime`: marca temporal.

  La marca temporal del momento en que se creó esta definición de conexión.

- `LastUpdatedTime`: marca temporal.

  La marca temporal de la última vez que se actualizó la definición de conexión.

- `LastUpdatedBy`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El usuario, el grupo o el rol que actualizaron por última vez la definición de esta conexión.

- `Status`: cadena UTF-8 (valores válidos: `READY` | `IN_PROGRESS` | `FAILED`).

  El estado de la conexión. Puede ser: `READY`, `IN_PROGRESS` o `FAILED`.

- `StatusReason`: Cadena UTF-8 con 1 byte de largo como mínimo o 16 384 como máximo.

  Motivo del estado de la conexión.

- `LastConnectionValidationTime`: marca temporal.

  Una marca temporal de la última vez que se validó esta conexión.

- `AuthenticationConfiguration`: un objeto AuthenticationConfiguration.

  Las propiedades de autenticación de la conexión.

Estructura ConnectionInput

Una estructura que se usa para especificar una conexión que se va a crear o a actualizar.

Campos

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la conexión.

- `Description`: cadena de descripción de un máximo de 2048 bytes de largo, que coincide con el URI address multi-line string pattern.

  La descripción de la conexión.

- `ConnectionType`: obligatorio: Cadena UTF-8 (valores válidos: `JDBC` | `SFTP` | `MONGODB` | `KAFKA` | `NETWORK` | `MARKETPLACE` | `CUSTOM` | `SALESFORCE`).

  Tipo de la conexión. Actualmente, estos tipos son compatibles:

- JDBC: designa una conexión a una base de datos a través de Java Database Connectivity (JDBC).

  Las conexiones JDBC utilizan los siguientes parámetros ConnectionParameters.

  - Obligatorios: todos estos: HOST, PORT, JDBC_ENGINE; o bien JDBC_CONNECTION_URL.
  - Obligatorios: todos estos: USERNAME, PASSWORD; o bien SECRET_ID.
  - Opcionales: JDBC_ENFORCE_SSL, CUSTOM_JDBC_CERT, CUSTOM_JDBC_CERT_STRING, SKIP_CUSTOM_JDBC_CERT_VALIDATION. Estos parámetros se utilizan para configurar SSL con JDBC.

- KAFKA: designa una conexión a una plataforma de streaming Apache Kafka.

  Las conexiones KAFKA utilizan los siguientes parámetros ConnectionParameters.

  - Obligatorio: KAFKA_BOOTSTRAP_SERVERS.
  - Opcionales: KAFKA_SSL_ENABLED, KAFKA_CUSTOM_CERT, KAFKA_SKIP_CUSTOM_CERT_VALIDATION. Estos parámetros se utilizan para configurar SSL con KAFKA.
  - Opcionales: KAFKA_CLIENT_KEYSTORE, KAFKA_CLIENT_KEYSTORE_PASSWORD, KAFKA_CLIENT_KEY_PASSWORD, ENCRYPTED_KAFKA_CLIENT_KEYSTORE_PASSWORD, ENCRYPTED_KAFKA_CLIENT_KEY_PASSWORD. Estos parámetros se utilizan para establecer la configuración del cliente TLS con SSL en KAFKA.
  - Opcional: KAFKA_SASL_MECHANISM. Se puede especificar como SCRAM-SHA-512, GSSAPI o AWS_MSK_IAM.
  - Opcionales: KAFKA_SASL_SCRAM_USERNAME, KAFKA_SASL_SCRAM_PASSWORD, ENCRYPTED_KAFKA_SASL_SCRAM_PASSWORD. Estos parámetros se utilizan para configurar la autenticación SASL/SCRAM-SHA-512 con KAFKA.
  - Opcionales: KAFKA_SASL_GSSAPI_KEYTAB, KAFKA_SASL_GSSAPI_KRB5_CONF, KAFKA_SASL_GSSAPI_SERVICE, KAFKA_SASL_GSSAPI_PRINCIPAL. Estos parámetros se utilizan para configurar la autenticación SASL/GSSAPI con KAFKA.

- MONGODB: designa una conexión a una base de datos de documentos MongoDB.

  Las conexiones MONGODB utilizan los siguientes parámetros ConnectionParameters.

  - Obligatorio: CONNECTION_URL.
  - Obligatorios: todos estos: USERNAME, PASSWORD; o bien SECRET_ID.

- SALESFORCE: Designa una conexión a Salesforce mediante el uso de la autenticación de OAuth.

  - Requiere que se configure el miembro `AuthenticationConfiguration`.
- `NETWORK`: designa una conexión de red a un origen de datos dentro de un entorno de Amazon Virtual Private Cloud (Amazon VPC).

  Las conexiones `NETWORK` no requieren parámetros ConnectionParameters. En su lugar, proporcione un requisito PhysicalConnectionRequirements.
- `MARKETPLACE`: utiliza los ajustes de configuración contenidos en un conector adquirido de AWS Marketplace para leer y escribir en almacenes de datos que no son soportados de forma nativa por AWS Glue.

  Las conexiones `MARKETPLACE` utilizan los siguientes parámetros ConnectionParameters.
  - Obligatorios: `CONNECTOR_TYPE`, `CONNECTOR_URL`, `CONNECTOR_CLASS_NAME`, `CONNECTION_URL`.
  - Obligatorios para conexiones `JDBC` `CONNECTOR_TYPE`: todos estos: `USERNAME`, `PASSWORD`; o bien `SECRET_ID`.
- `CUSTOM`: utiliza los ajustes de configuración contenidos en un conector personalizado para leer y escribir en almacenes de datos que no son soportados de forma nativa por AWS Glue.

  `SFTP` no se admite.

  Para obtener más información sobre cómo se pueden utilizar propiedades ConnectionProperties opcionales para configurar características en AWS Glue, consulte Propiedades de las conexiones de AWS Glue.

  Para obtener más información sobre cómo se pueden utilizar propiedades ConnectionProperties opcionales para configurar características en AWS Glue Studio, consulte Uso de conectores y conexiones.
- `MatchCriteria`: matriz de cadenas UTF-8, con 10 cadenas como máximo.

  Una lista de criterios que se pueden usar para seleccionar esta conexión.
- `ConnectionProperties` – Obligatorio: matriz de mapas de pares clave-valor, con 100 pares como máximo.

  Cada clave es una cadena UTF-8 (valores válidos: `HOST` | `PORT` | `USERNAME="USER_NAME"` | `PASSWORD` | `ENCRYPTED_PASSWORD` | `JDBC_DRIVER_JAR_URI` | `JDBC_DRIVER_CLASS_NAME` | `JDBC_ENGINE` | `JDBC_ENGINE_VERSION` | `CONFIG_FILES` | `INSTANCE_ID` | `JDBC_CONNECTION_URL` | `JDBC_ENFORCE_SSL` | `CUSTOM_JDBC_CERT` |

SKIP_CUSTOM_JDBC_CERT_VALIDATION | CUSTOM_JDBC_CERT_STRING | CONNECTION_URL | KAFKA_BOOTSTRAP_SERVERS | KAFKA_SSL_ENABLED | KAFKA_CUSTOM_CERT | KAFKA_SKIP_CUSTOM_CERT_VALIDATION | KAFKA_CLIENT_KEYSTORE | KAFKA_CLIENT_KEYSTORE_PASSWORD | KAFKA_CLIENT_KEY_PASSWORD | ENCRYPTED_KAFKA_CLIENT_KEYSTORE_PASSWORD | ENCRYPTED_KAFKA_CLIENT_KEY_PASSWORD | SECRET_ID | CONNECTOR_URL | CONNECTOR_TYPE | CONNECTOR_CLASS_NAME | KAFKA_SASL_MECHANISM | KAFKA_SASL_PLAIN_USERNAME | KAFKA_SASL_PLAIN_PASSWORD | ENCRYPTED_KAFKA_SASL_PLAIN_PASSWORD | KAFKA_SASL_SCRAM_USERNAME | KAFKA_SASL_SCRAM_PASSWORD | KAFKA_SASL_SCRAM_SECRETS_ARN | ENCRYPTED_KAFKA_SASL_SCRAM_PASSWORD | KAFKA_SASL_GSSAPI_KEYTAB | KAFKA_SASL_GSSAPI_KRB5_CONF | KAFKA_SASL_GSSAPI_SERVICE | KAFKA_SASL_GSSAPI_PRINCIPAL | ROLE_ARN).

Cada valor es una cadena de valor que no tiene más de 1024 bytes de largo.

Estos pares de clave-valor definen parámetros para la conexión.

- `PhysicalConnectionRequirements`: un objeto PhysicalConnectionRequirements.

  Los requisitos de conexión física, como la nube privada virtual (VPC) y `SecurityGroup`, que se necesitan para crear esta conexión correctamente.

- `AuthenticationConfiguration`: un objeto AuthenticationConfigurationInput.

  Las propiedades de autenticación de la conexión. Se usa para una conexión de Salesforce.

- `ValidateCredentials`: booleano.

  Un indicador para validar las credenciales durante la creación de la conexión. Se usa para una conexión de Salesforce. El valor predeterminado es verdadero.

Estructura PhysicalConnectionRequirements

La aplicación cliente de OAuth en la respuesta GetConnection.

Campos

- `SubnetId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

El ID de subred utilizado por la conexión.

- `SecurityGroupIdList`: matriz de cadenas UTF-8, con 50 cadenas como máximo.

  La lista de ID de grupo de seguridad usados por la conexión.

- `AvailabilityZone`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  La zona de disponibilidad de la conexión.

Estructura GetConnectionsFilter

Filtra las definiciones de conexión que devuelve la operación de la API `GetConnections`.

Campos

- `MatchCriteria`: matriz de cadenas UTF-8, con 10 cadenas como máximo.

  Una cadena de criterios que debe coincidir con los criterios registrados en la definición de la conexión para que dicha definición se devuelva.

- `ConnectionType`: cadena UTF-8 (valores válidos: `JDBC` | `SFTP` | `MONGODB` | `KAFKA` | `NETWORK` | `MARKETPLACE` | `CUSTOM` | `SALESFORCE`).

  Tipo de conexiones que se van a devolver. Actualmente, SFTP no es compatible.

Operaciones

- Acción CreateConnection (Python: create_connection)
- Acción DeleteConnection (Python: delete_connection)
- Acción GetConnection (Python: get_connection)
- Acción GetConnections (Python: get_connections)
- Acción UpdateConnection (Python: update_connection)
- Acción BatchDeleteConnection (Python: batch_delete_connection)

Acción CreateConnection (Python: create_connection)

Crea una definición de la conexión en el Catálogo de datos.

Las conexiones utilizadas para crear recursos federados requieren el permiso `glue:PassConnection` de IAM.

Solicitud

- `CatalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID del Catálogo de datos donde se crea la conexión. Si no se proporciona ninguno, se usará de forma predeterminada el ID de cuenta de AWS.
- `ConnectionInput`: obligatorio: objeto ConnectionInput.

  Objeto `ConnectionInput` que define la conexión que se va a crear.
- `Tags`: matriz de mapas de pares de clave-valor, con 50 pares como máximo.

  Cada clave es una cadena UTF-8 con una longitud de entre 1 y 128 bytes.

  Cada valor es una cadena UTF-8 que no tiene más de 256 bytes de largo.

  Etiquetas que asigna a la conexión.

Respuesta

- `CreateConnectionStatus`: cadena UTF-8 (valores válidos: `READY` | `IN_PROGRESS` | `FAILED`).

  El estado de la solicitud de creación de conexión. La solicitud puede llevar tiempo en algunos tipos de autenticación, por ejemplo, cuando se crea una conexión OAuth con intercambio de tokens por medio de una VPC.

Errores

- `AlreadyExistsException`
- `InvalidInputException`
- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`
- `GlueEncryptionException`

Acción DeleteConnection (Python: delete_connection)

Elimina una conexión del Catálogo de datos.

Solicitud

- `CatalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID del Catálogo de datos donde reside la conexión. Si no se proporciona ninguno, se usará de forma predeterminada el ID de cuenta de AWS.
- `ConnectionName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la conexión que se va a eliminar.

Respuesta

- Sin parámetros de respuesta.

Errores

- `EntityNotFoundException`
- `OperationTimeoutException`

Acción GetConnection (Python: get_connection)

Recupera una definición de la conexión del Catálogo de datos.

Solicitud

- `CatalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID del Catálogo de datos donde reside la conexión. Si no se proporciona ninguno, se usará de forma predeterminada el ID de cuenta de AWS.
- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

El nombre de la definición de la conexión que se va a recuperar.

- `HidePassword`: booleano.

  Le permite recuperar los metadatos de conexión sin devolver la contraseña. Por ejemplo, la consola de AWS Glue utiliza esta marca para recuperar la conexión y no muestra la contraseña. Establezca este parámetro cuando es posible que el que realiza la llamada no tenga permiso para usar la clave AWS KMS para descifrar la contraseña, pero tenga permiso para acceder al resto de las propiedades de conexión.

Respuesta

- `Connection`: un objeto Connection.

  La definición de la conexión solicitada.

Errores

- `EntityNotFoundException`
- `OperationTimeoutException`
- `InvalidInputException`
- `GlueEncryptionException`

Acción GetConnections (Python: get_connections)

Recupera una lista de definiciones de la conexión del Catálogo de datos.

Solicitud

- `CatalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID del Catálogo de datos donde residen las conexiones. Si no se proporciona ninguno, se usará de forma predeterminada el ID de cuenta de AWS.

- `Filter`: un objeto GetConnectionsFilter.

  Un filtro que controla las conexiones que se devolverán.

- `HidePassword`: booleano.

Le permite recuperar los metadatos de conexión sin devolver la contraseña. Por ejemplo, la consola de AWS Glue utiliza esta marca para recuperar la conexión y no muestra la contraseña. Establezca este parámetro cuando es posible que el que realiza la llamada no tenga permiso para usar la clave AWS KMS para descifrar la contraseña, pero tenga permiso para acceder al resto de las propiedades de conexión.

- `NextToken`: cadena UTF-8.

  Token de continuación si se trata de una llamada de continuidad.

- `MaxResults`: número (entero) que no es inferior a 1 ni es superior a 1000.

  El número máximo de conexiones que se devuelven en una respuesta.

Respuesta

- `ConnectionList`: matriz de objetos Connection.

  Una lista de definiciones de la conexión solicitadas.

- `NextToken`: cadena UTF-8.

  Un token de continuación, si la lista de conexiones que se devuelve no incluye la última conexión filtrada.

Errores

- `EntityNotFoundException`
- `OperationTimeoutException`
- `InvalidInputException`
- `GlueEncryptionException`

Acción UpdateConnection (Python: update_connection)

Actualiza una definición de la conexión en el Catálogo de datos.

Solicitud

- `CatalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

El ID del Catálogo de datos donde reside la conexión. Si no se proporciona ninguno, se usará de forma predeterminada el ID de cuenta de AWS.

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la definición de la conexión que se va a actualizar.

- `ConnectionInput`: obligatorio: objeto ConnectionInput.

  Un objeto `ConnectionInput` que redefine la conexión en cuestión.

Respuesta

- Sin parámetros de respuesta.

Errores

- `InvalidInputException`
- `EntityNotFoundException`
- `OperationTimeoutException`
- `InvalidInputException`
- `GlueEncryptionException`

Acción BatchDeleteConnection (Python: batch_delete_connection)

Elimina una lista de definiciones de la conexión del Catálogo de datos.

Solicitud

- `CatalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID del Catálogo de datos donde residen las conexiones. Si no se proporciona ninguno, se usará de forma predeterminada el ID de cuenta de AWS.

- `ConnectionNameList` – Obligatorio: matriz de cadenas UTF-8, con 25 cadenas como máximo.

  Una lista de nombres de las conexiones que se van a eliminar.

Respuesta

- `Succeeded`: matriz de cadenas UTF-8.

  Una lista de nombres de las definiciones de la conexión que se eliminaron con éxito.
- `Errors`: matriz de mapas de pares clave-valor.

  Cada clave es una cadena UTF-8 con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Cada valor es un objeto ErrorDetail.

  Una asignación de los nombres de conexiones que no se eliminaron con éxito a los detalles del error.

Errores

- `InternalServiceException`
- `OperationTimeoutException`

Configuración de la autenticación

- Estructura AuthenticationConfiguration
- Estructura AuthenticationConfigurationInput
- Estructura OAuth2Properties
- Estructura OAuth2PropertiesInput
- Estructura OAuth2ClientApplication
- Estructura AuthorizationCodeProperties

Estructura AuthenticationConfiguration

Estructura que contiene la configuración de la autenticación.

Campos

- `AuthenticationType`: cadena UTF-8 (valores válidos: `BASIC` | `OAUTH2` | `CUSTOM`).

  Estructura que contiene la configuración de la autenticación.

- `SecretArn`: cadena UTF-8 que coincide con el Custom string pattern #11.

  El ARN de Secrets Manager para almacenar las credenciales.
- `OAuth2Properties`: un objeto OAuth2Properties.

  Las propiedades para la autenticación de OAuth2.

Estructura AuthenticationConfigurationInput

Estructura que contiene la configuración de la autenticación en la solicitud CreateConnection.

Campos

- `AuthenticationType`: cadena UTF-8 (valores válidos: `BASIC` | `OAUTH2` | `CUSTOM`).

  Estructura que contiene la configuración de la autenticación en la solicitud CreateConnection.
- `SecretArn`: cadena UTF-8 que coincide con el Custom string pattern #11.

  El ARN de Secrets Manager para almacenar las credenciales en la solicitud CreateConnection.
- `OAuth2Properties`: un objeto OAuth2PropertiesInput.

  Las propiedades de la autenticación de OAuth2 en la solicitud CreateConnection.

Estructura OAuth2Properties

Una estructura que contiene propiedades para la autenticación de OAuth2.

Campos

- `OAuth2GrantType`: cadena UTF-8 (valores válidos: `AUTHORIZATION_CODE` | `CLIENT_CREDENTIALS` | `JWT_BEARER`).

  El tipo de concesión de OAuth2. Por ejemplo, `AUTHORIZATION_CODE`, `JWT_BEARER` o `CLIENT_CREDENTIALS`.
- `OAuth2ClientApplication`: un objeto OAuth2ClientApplication.

  El tipo de aplicación cliente. Por ejemplo, AWS_MANAGED o USER_MANAGED.
- `TokenUrl`: cadena UTF-8, de 256 bytes de largo como máximo, que coincide con Custom string pattern #12.

La URL del servidor de autenticación del proveedor para intercambiar un código de autorización por un token de acceso.

- `TokenUrlParametersMap`: matriz de mapas de pares clave-valor.

  Cada clave es una cadena UTF-8 con una longitud de entre 1 y 128 bytes.

  Cada valor es una cadena UTF-8 con 1 byte de largo como mínimo o 512 como máximo.

  Un mapa de parámetros que se añaden a la solicitud `GET` de token.

Estructura OAuth2PropertiesInput

Estructura que contiene las propiedades para OAuth2 en la solicitud CreateConnection.

Campos

- `OAuth2GrantType`: cadena UTF-8 (valores válidos: `AUTHORIZATION_CODE` | `CLIENT_CREDENTIALS` | `JWT_BEARER`).

  El tipo de concesión de OAuth2 en la solicitud CreateConnection. Por ejemplo, `AUTHORIZATION_CODE`, `JWT_BEARER` o `CLIENT_CREDENTIALS`.

- `OAuth2ClientApplication`: un objeto OAuth2ClientApplication.

  El tipo de aplicación cliente en la solicitud CreateConnection. Por ejemplo, `AWS_MANAGED` o `USER_MANAGED`.

- `TokenUrl`: cadena UTF-8, de 256 bytes de largo como máximo, que coincide con Custom string pattern #12.

  La URL del servidor de autenticación del proveedor para intercambiar un código de autorización por un token de acceso.

- `TokenUrlParametersMap`: matriz de mapas de pares clave-valor.

  Cada clave es una cadena UTF-8 con una longitud de entre 1 y 128 bytes.

  Cada valor es una cadena UTF-8 con 1 byte de largo como mínimo o 512 como máximo.

  Un mapa de parámetros que se añaden a la solicitud `GET` de token.

- `AuthorizationCodeProperties`: un objeto AuthorizationCodeProperties.

El conjunto de propiedades necesarias para el tipo de concesión `AUTHORIZATION_CODE` de OAuth2.

Estructura OAuth2ClientApplication

La aplicación cliente de OAuth2 que se usa para la conexión.

Campos

- `UserManagedClientApplicationClientId`: cadena UTF-8 con un máximo de 2048 bytes de largo, que coincide con el Custom string pattern #13.

  El ClientID de la aplicación cliente si el tipo de aplicación cliente es `USER_MANAGED`.
- `AWSManagedClientApplicationReference`: cadena UTF-8 con un máximo de 2048 bytes de largo, que coincide con el Custom string pattern #13.

  La referencia a la aplicación cliente del lado SaaS administrada por AWS.

Estructura AuthorizationCodeProperties

El conjunto de propiedades necesario para el flujo de trabajo del tipo de concesión `AUTHORIZATION_CODE` de OAuth2.

Campos

- `AuthorizationCode`: Cadena UTF-8 con 1 byte de largo como mínimo o 4096 como máximo, que coincide con Custom string pattern #13.

  Un código de autorización que se usará en la tercera fase del flujo de trabajo de la concesión `AUTHORIZATION_CODE`. Se trata de un código de un solo uso que deja de ser válido una vez que se intercambia por un token de acceso, por lo que es aceptable tener este valor como parámetro de solicitud.
- `RedirectUri`: Cadena UTF-8 con 512 bytes de largo como máximo, que coincide con Custom string pattern #14.

  El URI de redireccionamiento al que el servidor de autorización redirige al usuario al emitir un código de autorización. El URI se usa posteriormente cuando el código de autorización se intercambia por un token de acceso.

API de funciones definidas por el usuario

La API de funciones definidas por el usuario describe los tipos de datos y las operaciones de AWS Glue que se utilizan para trabajar con funciones.

Tipos de datos

- Estructura UserDefinedFunction
- Estructura UserDefinedFunctionInput

Estructura UserDefinedFunction

Representa el equivalente a una definición de función Hive definida por el usuario (UDF).

Campos

- `FunctionName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la función.
- `DatabaseName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre de la base de datos del catálogo que contiene la función.
- `ClassName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  La clase de Java que contiene el código de la función.
- `OwnerName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El propietario de la función.
- `OwnerType`: cadena UTF-8 (valores válidos: `USER` | `ROLE` | `GROUP`).

  El tipo de propietario.
- `CreateTime`: marca temporal.

  La hora a la que se creó la función.

- `ResourceUris`: matriz de objetos ResourceUri, con 1000 estructuras como máximo.

  Los URI del recurso para la función.
- `CatalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID del Catálogo de datos donde reside la función.

Estructura UserDefinedFunctionInput

Un estructura que se utiliza para crear o actualizar funciones definidas por el usuario.

Campos

- `FunctionName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la función.
- `ClassName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  La clase de Java que contiene el código de la función.
- `OwnerName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El propietario de la función.
- `OwnerType`: cadena UTF-8 (valores válidos: `USER` | `ROLE` | `GROUP`).

  El tipo de propietario.
- `ResourceUris`: matriz de objetos ResourceUri, con 1000 estructuras como máximo.

  Los URI del recurso para la función.

Operaciones

- Acción CreateUserDefinedFunction (Python: create_user_defined_function)
- Acción UpdateUserDefinedFunction (Python: update_user_defined_function)
- Acción DeleteUserDefinedFunction (Python: delete_user_defined_function)

- Acción GetUserDefinedFunction (Python: get_user_defined_function)
- Acción GetUserDefinedFunctions (Python: get_user_defined_functions)

Acción CreateUserDefinedFunction (Python: create_user_defined_function)

Crea una nueva definición de función en el Catálogo de datos.

Solicitud

- `CatalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID del Catálogo de datos donde se crea la función. Si no se proporciona ninguno, se usará de forma predeterminada el ID de cuenta de AWS.
- `DatabaseName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la base de datos de catálogos donde se crea la función.
- `FunctionInput` – Obligatorio: objeto UserDefinedFunctionInput.

  Objeto `FunctionInput` que define la función que se va a crear en el Catálogo de datos.

Respuesta

- Sin parámetros de respuesta.

Errores

- `AlreadyExistsException`
- `InvalidInputException`
- `InternalServiceException`
- `EntityNotFoundException`
- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`
- `GlueEncryptionException`

Acción UpdateUserDefinedFunction (Python: update_user_defined_function)

Actualiza una definición de función existente en el Catálogo de datos.

Solicitud

- `CatalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID del Catálogo de datos donde se encuentra la función que se va a actualizar. Si no se proporciona ninguno, se usará de forma predeterminada el ID de cuenta de AWS.
- `DatabaseName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la base de datos de catálogos donde se encuentra la función que se va a actualizar.
- `FunctionName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la función.
- `FunctionInput` – Obligatorio: objeto UserDefinedFunctionInput.

  Un objeto `FunctionInput` que redefine la función que en el Catálogo de datos.

Respuesta

- Sin parámetros de respuesta.

Errores

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`

Acción DeleteUserDefinedFunction (Python: delete_user_defined_function)

Elimina una definición de función existente del Catálogo de datos.

Solicitud

- `CatalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID del Catálogo de datos donde se encuentra la función que se va a eliminar. Si no se proporciona ninguno, el ID de cuenta de AWS se usará de forma predeterminada.
- `DatabaseName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la base de datos de catálogos donde se encuentra la función.
- `FunctionName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la definición de la función que se va a eliminar.

Respuesta

- Sin parámetros de respuesta.

Errores

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

Acción GetUserDefinedFunction (Python: get_user_defined_function)

Recupera una definición de función especificada del Catálogo de datos.

Solicitud

- `CatalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID del Catálogo de datos donde se encuentra la función que se va a recuperar. Si no se proporciona ninguno, se usará de forma predeterminada el ID de cuenta de AWS.
- `DatabaseName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la base de datos de catálogos donde se encuentra la función.
- `FunctionName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la función.

Respuesta

- `UserDefinedFunction`: un objeto UserDefinedFunction.

  La definición de la función solicitada.

Errores

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `GlueEncryptionException`

Acción GetUserDefinedFunctions (Python: get_user_defined_functions)

Recupera varias definiciones de función del Catálogo de datos.

Solicitud

- `CatalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

El ID del Catálogo de datos donde se encuentran las funciones que se van a recuperar. Si no se proporciona ninguno, se usará de forma predeterminada el ID de cuenta de AWS.

- `DatabaseName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la base de datos de catálogos donde se encuentra las funciones. Si no se proporciona ninguno, se devolverán las funciones de todas las bases de datos del catálogo.
- `Pattern`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Cadena con patrón nombre-de-función opcional que filtra las definiciones de función que se devuelven.
- `NextToken`: cadena UTF-8.

  Token de continuación si se trata de una llamada de continuidad.
- `MaxResults`: número (entero) que no es inferior a 1 ni es superior a 100.

  El número máximo de funciones que se devuelven en una respuesta.

Respuesta

- `UserDefinedFunctions`: matriz de objetos UserDefinedFunction.

  Lista de las definiciones de función solicitadas.
- `NextToken`: cadena UTF-8.

  Token de continuación, si la lista de funciones que se devuelve no incluye la última función solicitada.

Errores

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`
- `GlueEncryptionException`

Importación de un catálogo de Athena a AWS Glue

La API de migración describe los tipos de datos y las operaciones de AWS Glue relativos a la migración de un Catálogo de datos de Athena a AWS Glue.

Tipos de datos

- Estructura CatalogImportStatus

Estructura CatalogImportStatus

Estructura que contiene información sobre el estado de la migración.

Campos

- `ImportCompleted`: booleano.

  `True` si la migración se ha completado; o `False` en caso contrario,.
- `ImportTime`: marca temporal.

  La hora en que comenzó la migración.
- `ImportedBy`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la persona que inició la migración.

Operaciones

- Acción ImportCatalogToGlue (Python: import_catalog_to_glue)
- Acción GetCatalogImportStatus (Python: get_catalog_import_status)

Acción ImportCatalogToGlue (Python: import_catalog_to_glue)

Importa un Catálogo de datos de Amazon Athena existente a AWS Glue.

Solicitud

- `CatalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

El ID del catálogo que se va a importar. En la actualidad, debe ser el ID de cuenta de AWS.

Respuesta

- Sin parámetros de respuesta.

Errores

- `InternalServiceException`
- `OperationTimeoutException`

Acción GetCatalogImportStatus (Python: get_catalog_import_status)

Recupera el estado de una operación de migración.

Solicitud

- `CatalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID del catálogo que se va a migrar. En la actualidad, debe ser el ID de cuenta de AWS.

Respuesta

- `ImportStatus`: un objeto CatalogImportStatus.

  El estado de la migración del catálogo especificado.

Errores

- `InternalServiceException`
- `OperationTimeoutException`

API del optimizador de tablas

La API del optimizador de tablas describe la AWS Glue API que permite la compactación para mejorar el rendimiento de lectura.

Tipos de datos

- TableOptimizer estructura
- TableOptimizerConfiguration estructura
- TableOptimizerRun estructura
- RunMetrics estructura
- BatchGetTableOptimizerEntry estructura
- BatchTableOptimizer estructura
- BatchGetTableOptimizerError estructura

TableOptimizer estructura

Contiene detalles sobre un optimizador asociado a una tabla.

Campos

- `type`: cadena UTF-8 (valores válidos: `compaction="COMPACTION"`).

  El tipo de optimizador de tabla. El único valor válido actualmente es `compaction`.
- `configuration`: un objeto TableOptimizerConfiguration.

  Un objeto `TableOptimizerConfiguration` que se especificó al crear o actualizar un optimizador de tablas.
- `lastRun`: un objeto TableOptimizerRun.

  Un objeto `TableOptimizerRun` que representa la última ejecución del optimizador de tablas.

TableOptimizerConfiguration estructura

Contiene detalles sobre la configuración de un optimizador de tablas. Esta configuración se transfiere al crear o actualizar un optimizador de tablas.

Campos

- `roleArn`: cadena UTF-8, con 1 byte de largo como mínimo y 512 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Un rol transferido por la persona que llama y que permite al servicio actualizar los recursos asociados al optimizador en nombre de la persona que llama.
- `enabled`: booleano.

  Si la optimización de tablas está habilitada.

TableOptimizerRun estructura

Contiene detalles de una ejecución del optimizador de tablas.

Campos

- `eventType`: cadena UTF-8 (valores válidos: `starting="STARTING"` | `completed="COMPLETED"` | `failed="FAILED"` | `in_progress="IN_PROGRESS"`).

  Un tipo de evento que representa el estado de la ejecución del optimizador de tablas.
- `startTimestamp`: marca temporal.

  Representa la marca temporal de la época en la que se inició el trabajo de compactación en Lake Formation.
- `endTimestamp`: marca temporal.

  Representa la marca temporal de la época en la que finalizó el trabajo de compactación.
- `metrics`: un objeto RunMetrics.

  Un objeto `RunMetrics` que contiene las métricas de la ejecución del optimizador.
- `error`: cadena UTF-8.

  Un error que se produjo durante la ejecución del optimizador.

RunMetrics estructura

Métricas de la ejecución del optimizador.

Campos

- `NumberOfBytesCompacted`: cadena UTF-8.

  El número de bytes eliminados por la ejecución del trabajo de compactación.
- `NumberOfFilesCompacted`: cadena UTF-8.

  El número de archivos eliminados por la ejecución del trabajo de compactación.
- `NumberOfDpus`: cadena UTF-8.

  El número de horas de DPU consumidas por el trabajo.
- `JobDurationInHour`: cadena UTF-8.

  La duración del trabajo en horas.

BatchGetTableOptimizerEntry estructura

Representa un optimizador de tablas que se va a recuperar en la operación `BatchGetTableOptimizer`.

Campos

- `catalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID de catálogo de la tabla.
- `databaseName`: cadena UTF-8 de 1 byte de largo como mínimo.

  El nombre de la base de datos en el catálogo donde reside la tabla.
- `tableName`: cadena UTF-8 de 1 byte de largo como mínimo.

  El nombre de la tabla.
- `type`: cadena UTF-8 (valores válidos: `compaction="COMPACTION"`).

  El tipo de optimizador de tabla.

BatchTableOptimizer estructura

Contiene detalles de uno de los optimizadores de tablas devueltos por la operación `BatchGetTableOptimizer`.

Campos

- `catalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID de catálogo de la tabla.
- `databaseName`: cadena UTF-8 de 1 byte de largo como mínimo.

  El nombre de la base de datos en el catálogo donde reside la tabla.
- `tableName`: cadena UTF-8 de 1 byte de largo como mínimo.

  El nombre de la tabla.
- `tableOptimizer`: un objeto TableOptimizer.

  Un objeto `TableOptimizer` que contiene detalles sobre la configuración y la última ejecución de un optimizador de tabla.

BatchGetTableOptimizerError estructura

Contiene detalles sobre uno de los errores de la lista de errores devuelta por la operación `BatchGetTableOptimizer`.

Campos

- `error`: un objeto ErrorDetail.

  Un objeto `ErrorDetail` que contiene detalles de mensaje y código acerca del error.
- `catalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID de catálogo de la tabla.
- `databaseName`: cadena UTF-8 de 1 byte de largo como mínimo.

  El nombre de la base de datos en el catálogo donde reside la tabla.

- `tableName`: cadena UTF-8 de 1 byte de largo como mínimo.

  El nombre de la tabla.
- `type`: cadena UTF-8 (valores válidos: `compaction="COMPACTION"`).

  El tipo de optimizador de tabla.

Operaciones

- GetTableOptimizer acción (Python: get_table_optimizer)
- BatchGetTableOptimizer acción (Python: batch_get_table_optimizer)
- ListTableOptimizerRuns acción (Python: list_table_optimizer_runs)
- CreateTableOptimizer acción (Python: create_table_optimizer)
- DeleteTableOptimizer acción (Python: delete_table_optimizer)
- UpdateTableOptimizer acción (Python: update_table_optimizer)

GetTableOptimizer acción (Python: get_table_optimizer)

Devuelve la configuración de todos los optimizadores asociados a una tabla especificada.

Solicitud

- `CatalogId`: Obligatorio: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID de catálogo de la tabla.
- `DatabaseName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la base de datos en el catálogo donde reside la tabla.
- `TableName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la tabla.
- `Type` – Obligatorio: cadena UTF-8 (valores válidos: `compaction="COMPACTION"`).

  El tipo de optimizador de tabla.

Respuesta

- `CatalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID de catálogo de la tabla.

- `DatabaseName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la base de datos en el catálogo donde reside la tabla.

- `TableName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la tabla.

- `TableOptimizer`: un objeto TableOptimizer.

  El optimizador asociado a la tabla especificada.

Errores

- `EntityNotFoundException`
- `InvalidInputException`
- `AccessDeniedException`
- `InternalServiceException`

BatchGetTableOptimizer acción (Python: batch_get_table_optimizer)

Devuelve la configuración de los optimizadores de la tabla especificados.

Solicitud

- `Entries` (obligatorio): una matriz de objetos BatchGetTableOptimizerEntry.

  Una lista de objetos `BatchGetTableOptimizerEntry` que especifican los optimizadores de tabla que se van a recuperar.

Respuesta

- `TableOptimizers`: matriz de objetos BatchTableOptimizer.

  Una lista de objetos `BatchTableOptimizer`.
- `Failures`: matriz de objetos BatchGetTableOptimizerError.

  Una lista de errores de la operación.

Errores

- `InternalServiceException`

ListTableOptimizerRuns acción (Python: list_table_optimizer_runs)

Muestra el historial de las ejecuciones anteriores del optimizador para una tabla específica.

Solicitud

- `CatalogId`: Obligatorio: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID de catálogo de la tabla.
- `DatabaseName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la base de datos en el catálogo donde reside la tabla.
- `TableName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la tabla.
- `Type` – Obligatorio: cadena UTF-8 (valores válidos: `compaction="COMPACTION"`).

  El tipo de optimizador de tabla. El único valor válido actualmente es `compaction`.
- `MaxResults`: número (entero).

  El número máximo de ejecuciones del optimizador que se devolverán en cada llamada.
- `NextToken`: cadena UTF-8.

Token de continuación si se trata de una llamada de continuidad.

Respuesta

- `CatalogId`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID de catálogo de la tabla.
- `DatabaseName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la base de datos en el catálogo donde reside la tabla.
- `TableName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la tabla.
- `NextToken`: cadena UTF-8.

  Token de continuación para paginar la lista obtenida; se devuelve si el segmento actual de la lista no es el último.
- `TableOptimizerRuns`: matriz de objetos TableOptimizerRun.

  Lista de las ejecuciones del optimizador asociadas a una tabla.

Errores

- `EntityNotFoundException`
- `AccessDeniedException`
- `InvalidInputException`
- `InternalServiceException`

CreateTableOptimizer acción (Python: create_table_optimizer)

Crea un nuevo optimizador de tablas para una función específica. `compaction` es el único tipo de optimizador compatible actualmente.

Solicitud

- `CatalogId`: Obligatorio: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID de catálogo de la tabla.
- `DatabaseName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la base de datos en el catálogo donde reside la tabla.
- `TableName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la tabla.
- `Type` – Obligatorio: cadena UTF-8 (valores válidos: `compaction="COMPACTION"`).

  El tipo de optimizador de tabla. El único valor válido actualmente es `compaction`.
- `TableOptimizerConfiguration`: obligatorio: objeto TableOptimizerConfiguration.

  Un objeto `TableOptimizerConfiguration` que representa la configuración de un optimizador de tablas.

Respuesta

- Sin parámetros de respuesta.

Errores

- `EntityNotFoundException`
- `InvalidInputException`
- `AccessDeniedException`
- `AlreadyExistsException`
- `InternalServiceException`

DeleteTableOptimizer acción (Python: delete_table_optimizer)

Elimina un optimizador y todos los metadatos asociados a una tabla. La optimización ya no se realizará en la tabla.

Solicitud

- `CatalogId`: Obligatorio: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID de catálogo de la tabla.
- `DatabaseName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la base de datos en el catálogo donde reside la tabla.
- `TableName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la tabla.
- `Type` – Obligatorio: cadena UTF-8 (valores válidos: `compaction="COMPACTION"`).

  El tipo de optimizador de tabla.

Respuesta

- Sin parámetros de respuesta.

Errores

- `EntityNotFoundException`
- `InvalidInputException`
- `AccessDeniedException`
- `InternalServiceException`

UpdateTableOptimizer acción (Python: update_table_optimizer)

Actualiza la configuración de un optimizador de tablas existente.

Solicitud

- `CatalogId`: Obligatorio: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID de catálogo de la tabla.

- `DatabaseName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la base de datos en el catálogo donde reside la tabla.

- `TableName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la tabla.

- `Type` – Obligatorio: cadena UTF-8 (valores válidos: `compaction="COMPACTION"`).

  El tipo de optimizador de tabla. El único valor válido actualmente es `compaction`.

- `TableOptimizerConfiguration`: obligatorio: objeto TableOptimizerConfiguration.

  Un objeto `TableOptimizerConfiguration` que representa la configuración de un optimizador de tablas.

Respuesta

- Sin parámetros de respuesta.

Errores

- `EntityNotFoundException`
- `InvalidInputException`
- `AccessDeniedException`
- `InternalServiceException`

API de rastreadores y clasificadores

La API de rastreadores y clasificadores describe los tipos de datos de rastreadores y clasificadores de AWS Glue e incluye la API para crear, eliminar, actualizar y ver listas de rastreadores o clasificadores.

Temas

- API de clasificador
- La API del rastreador
- API de estadísticas de columna
- API del programador del rastreador

API de clasificador

La API de clasificador describe los tipos de datos de clasificador de AWS Glue e incluye la API de creación, eliminación, actualización y creación de listados de clasificadores.

Tipos de datos

- Estructura Classifier
- Estructura GrokClassifier
- Estructura XMLClassifier
- Estructura JsonClassifier
- Estructura CsvClassifier
- Estructura CreateGrokClassifierRequest
- Estructura UpdateGrokClassifierRequest
- Estructura CreateXMLClassifierRequest
- Estructura UpdateXMLClassifierRequest
- Estructura CreateJsonClassifierRequest
- Estructura UpdateJsonClassifierRequest
- Estructura CreateCsvClassifierRequest
- Estructura UpdateCsvClassifierRequest

Estructura Classifier

Los clasificadores se disparan durante una tarea de rastreo. Un clasificador comprueba si un determinado archivo está en un formato que puede administrar. En caso afirmativo, el clasificador crea un esquema en forma de un objeto `StructType` que coincida con formato de datos.

Puede utilizar los clasificadores estándar que suministra AWS Glue o puede escribir sus propios clasificadores para clasificar mejor los orígenes de datos y especificar los esquemas adecuados para usar con ellos. Una clasificador puede ser un clasificador `grok`, un clasificador `XML`, un clasificador `JSON` o un clasificador `CSV` personalizado, en función de lo que se especifique en uno de los campos del objeto `Classifier`.

Campos

- `GrokClassifier`: un objeto GrokClassifier.

  Un clasificador que utiliza `grok`.
- `XMLClassifier`: un objeto XMLClassifier.

  Un clasificador de contenido XML.
- `JsonClassifier`: un objeto JsonClassifier.

  Un clasificador de contenido JSON.
- `CsvClassifier`: un objeto CsvClassifier.

  Un clasificador de valores separados por comas (CSV).

Estructura GrokClassifier

Un clasificador que utiliza patrones `grok`.

Campos

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre del clasificador.
- `Classification` – Obligatorio: cadena UTF-8.

Identificador del formato de los datos que el clasificador correlaciona; por ejemplo, Twitter, JSON, registros de Omniture, etc.

- `CreationTime`: marca temporal.

  La hora de registro de este clasificador.

- `LastUpdated`: marca temporal.

  La hora de actualización de este clasificador.

- `Version`: número (largo).

  Versión de este clasificador.

- `GrokPattern` – Obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 2048 bytes de largo como máximo, que coincide con el A Logstash Grok string pattern.

  El patrón grok que este clasificador aplica a un almacén de datos. Para obtener más información, consulte los patrones integrados en Escritura de clasificadores personalizados.

- `CustomPatterns`: cadena UTF-8 con un máximo de 16000 bytes de largo, que coincide con el URI address multi-line string pattern.

  Patrones de grok personalizados opcionales definidos por este clasificador. Para obtener más información, consulte los patrones personalizados en Escritura de clasificadores personalizados.

Estructura XMLClassifier

Un clasificador de contenido `XML`.

Campos

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre del clasificador.

- `Classification` – Obligatorio: cadena UTF-8.

  Identificador del formato de datos que el clasificador correlaciona.

- `CreationTime`: marca temporal.

  La hora de registro de este clasificador.

- `LastUpdated`: marca temporal.

  La hora de actualización de este clasificador.

- `Version`: número (largo).

  Versión de este clasificador.

- `RowTag`: cadena UTF-8.

  La etiqueta XML que designa el elemento que contiene cada registro en un documento XML que se está analizando. Esto no puede identificar un elemento de cierre (cerrado por `/>`). Un elemento de fila vacío que solo contenga atributos puede analizarse siempre y cuando finalice con una etiqueta de cierre (por ejemplo, `<row item_a="A" item_b="B"></row>` es correcto, pero `<row item_a="A" item_b="B" />` no lo es).

Estructura JsonClassifier

Un clasificador de contenido `JSON`.

Campos

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre del clasificador.

- `CreationTime`: marca temporal.

  La hora de registro de este clasificador.

- `LastUpdated`: marca temporal.

  La hora de actualización de este clasificador.

- `Version`: número (largo).

  Versión de este clasificador.

- `JsonPath` – Obligatorio: cadena UTF-8.

  Una cadena `JsonPath` que define los datos JSON que el clasificador debe clasificar. AWS Glue soporta un subconjunto de operadores de JsonPath, tal y como se describe en Escritura de clasificadores personalizados de JsonPath.

Estructura CsvClassifier

Clasificador de contenido CSV personalizado.

Campos

- Name: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre del clasificador.

- CreationTime: marca temporal.

  La hora de registro de este clasificador.

- LastUpdated: marca temporal.

  La hora de actualización de este clasificador.

- Version: número (largo).

  Versión de este clasificador.

- Delimiter: cadena UTF-8, con 1 byte de largo como mínimo y 1 byte de largo como máximo, que coincide con el Custom string pattern #10.

  Símbolo personalizado que indica qué elemento va a separar cada entrada de columna en la fila.

- QuoteSymbol: cadena UTF-8, con 1 byte de largo como mínimo y 1 byte de largo como máximo, que coincide con el Custom string pattern #10.

  Símbolo personalizado que indica qué elemento va a combinar contenido en un valor de columna único. Debe ser distinto al delimitador de columnas.

- ContainsHeader: cadena UTF-8 (valores válidos: UNKNOWN | PRESENT | ABSENT).

  Indica si el archivo CSV contiene un encabezado.

- Header: matriz de cadenas UTF-8.

  Lista de cadenas que representan nombres de columnas.

- DisableValueTrimming: booleano.

  Indica que los valores no deben recortarse antes de identificar el tipo de valores de columna. El valor predeterminado es true.

- `AllowSingleColumn`: booleano.

  Permite procesar los archivos que contienen una sola columna.

- `CustomDatatypeConfigured`: booleano.

  Permite configurar el tipo de datos personalizado.

- `CustomDatatypes`: matriz de cadenas UTF-8.

  Lista de tipos de datos personalizados que incluyen "BINARIO", "BOOLEANO", "FECHA", "DECIMAL", "DOUBLE", "FLOAT", "INT", "LONG", "SHORT", "STRING", "TIMESTAMP".

- `Serde`: cadena UTF-8 (valores válidos: `OpenCSVSerDe` | `LazySimpleSerDe` | `None`).

  Establece el SerDe para procesar CSV en el clasificador, que se aplicará en el catálogo de datos. Los valores válidos son `OpenCSVSerDe`, `LazySimpleSerDe` y `None`. Puede especificar el valor `None` cuando desee que el rastreador realice la detección.

Estructura CreateGrokClassifierRequest

Especifica un clasificador `grok` para que `CreateClassifier` lo cree.

Campos

- `Classification` – Obligatorio: cadena UTF-8.

  Identificador del formato de los datos que el clasificador correlaciona; por ejemplo, Twitter, JSON, registros de Omniture, Amazon CloudWatch Logs, etc.

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre del nuevo clasificador.

- `GrokPattern` – Obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 2048 bytes de largo como máximo, que coincide con el A Logstash Grok string pattern.

  Patrón de grok que este clasificador utiliza.

- `CustomPatterns`: cadena UTF-8 con un máximo de 16000 bytes de largo, que coincide con el URI address multi-line string pattern.

  Patrones de grok personalizados opcionales utilizados por este clasificador.

Estructura UpdateGrokClassifierRequest

Especifica un clasificador de grok que debe actualizarse cuando se pase a `UpdateClassifier`.

Campos

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre del elemento `GrokClassifier`.
- `Classification`: cadena UTF-8.

  Identificador del formato de los datos que el clasificador correlaciona; por ejemplo, Twitter, JSON, registros de Omniture, Amazon CloudWatch Logs, etc.
- `GrokPattern`: cadena UTF-8, con 1 byte de largo como mínimo y 2048 bytes de largo como máximo, que coincide con el A Logstash Grok string pattern.

  Patrón de grok que este clasificador utiliza.
- `CustomPatterns`: cadena UTF-8 con un máximo de 16000 bytes de largo, que coincide con el URI address multi-line string pattern.

  Patrones de grok personalizados opcionales utilizados por este clasificador.

Estructura CreateXMLClassifierRequest

Especifica un clasificador de XML para que `CreateClassifier` lo cree.

Campos

- `Classification` – Obligatorio: cadena UTF-8.

  Identificador del formato de datos que el clasificador correlaciona.
- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre del clasificador.
- `RowTag`: cadena UTF-8.

  La etiqueta XML que designa el elemento que contiene cada registro en un documento XML que se está analizando. Esto no puede identificar un elemento de cierre (cerrado por `/>`). Un elemento

de fila vacío que solo contenga atributos puede analizarse siempre y cuando finalice con una etiqueta de cierre (por ejemplo, `<row item_a="A" item_b="B"></row>` es correcto, pero `<row item_a="A" item_b="B" />` no lo es).

Estructura UpdateXMLClassifierRequest

Especifica un clasificador de XML que debe actualizarse.

Campos

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre del clasificador.

- `Classification`: cadena UTF-8.

  Identificador del formato de datos que el clasificador correlaciona.

- `RowTag`: cadena UTF-8.

  La etiqueta XML que designa el elemento que contiene cada registro en un documento XML que se está analizando. Tenga en cuenta que no puede identificar un elemento que se autocierra (cerrado con `/>`). Un elemento de fila vacío que solo contenga atributos puede analizarse siempre y cuando finalice con una etiqueta de cierre (por ejemplo, `<row item_a="A" item_b="B"></row>` es correcto, pero `<row item_a="A" item_b="B" />` no lo es).

Estructura CreateJsonClassifierRequest

Especifica un clasificador de JSON para que `CreateClassifier` lo cree.

Campos

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre del clasificador.

- `JsonPath` – Obligatorio: cadena UTF-8.

Una cadena `JsonPath` que define los datos JSON que el clasificador debe clasificar. AWS Glue soporta un subconjunto de operadores de JsonPath, tal y como se describe en Escritura de clasificadores personalizados de JsonPath.

Estructura UpdateJsonClassifierRequest

Especifica un clasificador de JSON que debe actualizarse.

Campos

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre del clasificador.

- `JsonPath`: cadena UTF-8.

  Una cadena `JsonPath` que define los datos JSON que el clasificador debe clasificar. AWS Glue soporta un subconjunto de operadores de JsonPath, tal y como se describe en Escritura de clasificadores personalizados de JsonPath.

Estructura CreateCsvClassifierRequest

Especifica un clasificador CSV personalizado para que `CreateClassifier` lo cree.

Campos

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre del clasificador.

- `Delimiter`: cadena UTF-8, con 1 byte de largo como mínimo y 1 byte de largo como máximo, que coincide con el Custom string pattern #10.

  Símbolo personalizado que indica qué elemento va a separar cada entrada de columna en la fila.

- `QuoteSymbol`: cadena UTF-8, con 1 byte de largo como mínimo y 1 byte de largo como máximo, que coincide con el Custom string pattern #10.

Símbolo personalizado que indica qué elemento va a combinar contenido en un valor de columna único. Debe ser diferente al delimitador de columnas.

- `ContainsHeader`: cadena UTF-8 (valores válidos: `UNKNOWN` | `PRESENT` | `ABSENT`).

  Indica si el archivo CSV contiene un encabezado.

- `Header`: matriz de cadenas UTF-8.

  Lista de cadenas que representan nombres de columnas.

- `DisableValueTrimming`: booleano.

  Indica que los valores no deben recortarse antes de identificar el tipo de valores de columna. El valor predeterminado es true.

- `AllowSingleColumn`: booleano.

  Permite procesar los archivos que contienen una sola columna.

- `CustomDatatypeConfigured`: booleano.

  Permite la configuración de tipos de datos personalizados.

- `CustomDatatypes`: matriz de cadenas UTF-8.

  Crea una lista de tipos de datos personalizados compatibles.

- `Serde`: cadena UTF-8 (valores válidos: `OpenCSVSerDe` | `LazySimpleSerDe` | `None`).

  Establece el SerDe para procesar CSV en el clasificador, que se aplicará en el catálogo de datos. Los valores válidos son `OpenCSVSerDe`, `LazySimpleSerDe` y `None`. Puede especificar el valor `None` cuando desee que el rastreador realice la detección.

Estructura UpdateCsvClassifierRequest

Especifica un clasificador CSV personalizado para que se actualice.

Campos

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre del clasificador.

- `Delimiter`: cadena UTF-8, con 1 byte de largo como mínimo y 1 byte de largo como máximo, que coincide con el Custom string pattern #10.

  Símbolo personalizado que indica qué elemento va a separar cada entrada de columna en la fila.

- `QuoteSymbol`: cadena UTF-8, con 1 byte de largo como mínimo y 1 byte de largo como máximo, que coincide con el Custom string pattern #10.

  Símbolo personalizado que indica qué elemento va a combinar contenido en un valor de columna único. Debe ser distinto al delimitador de columnas.

- `ContainsHeader`: cadena UTF-8 (valores válidos: `UNKNOWN` | `PRESENT` | `ABSENT`).

  Indica si el archivo CSV contiene un encabezado.

- `Header`: matriz de cadenas UTF-8.

  Lista de cadenas que representan nombres de columnas.

- `DisableValueTrimming`: booleano.

  Indica que los valores no deben recortarse antes de identificar el tipo de valores de columna. El valor predeterminado es true.

- `AllowSingleColumn`: booleano.

  Permite procesar los archivos que contienen una sola columna.

- `CustomDatatypeConfigured`: booleano.

  Especifica la configuración de los tipos de datos personalizados.

- `CustomDatatypes`: matriz de cadenas UTF-8.

  Especifica una lista de tipos de datos personalizados compatibles.

- `Serde`: cadena UTF-8 (valores válidos: `OpenCSVSerDe` | `LazySimpleSerDe` | `None`).

  Establece el SerDe para procesar CSV en el clasificador, que se aplicará en el catálogo de datos. Los valores válidos son `OpenCSVSerDe`, `LazySimpleSerDe` y `None`. Puede especificar el valor `None` cuando desee que el rastreador realice la detección.

Operaciones

- Acción CreateClassifier (Python: create_classifier)

- Acción DeleteClassifier (Python: delete_classifier)
- Acción GetClassifier (Python: get_classifier)
- Acción GetClassifiers (Python: get_classifiers)
- Acción UpdateClassifier (Python: update_classifier)

Acción CreateClassifier (Python: create_classifier)

Crea un clasificador en la cuenta del usuario. Puede ser `GrokClassifier`, `XMLClassifier`, `JsonClassifier` o `CsvClassifier`, en función de qué campo de la solicitud esté presente.

Solicitud

- `GrokClassifier`: un objeto CreateGrokClassifierRequest.

  Objeto `GrokClassifier` que especifica el clasificador que debe crearse.
- `XMLClassifier`: un objeto CreateXMLClassifierRequest.

  Objeto `XMLClassifier` que especifica el clasificador que debe crearse.
- `JsonClassifier`: un objeto CreateJsonClassifierRequest.

  Objeto `JsonClassifier` que especifica el clasificador que debe crearse.
- `CsvClassifier`: un objeto CreateCsvClassifierRequest.

  Objeto `CsvClassifier` que especifica el clasificador que debe crearse.

Respuesta

- Sin parámetros de respuesta.

Errores

- `AlreadyExistsException`
- `InvalidInputException`
- `OperationTimeoutException`

Acción DeleteClassifier (Python: delete_classifier)

Quita un clasificador del catálogo de datos.

Solicitud

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre del clasificador que debe eliminarse.

Respuesta

- Sin parámetros de respuesta.

Errores

- `EntityNotFoundException`
- `OperationTimeoutException`

Acción GetClassifier (Python: get_classifier)

Recupera un clasificador por su nombre.

Solicitud

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre del clasificador que debe recuperarse.

Respuesta

- `Classifier`: un objeto Clasificador.

  El clasificador solicitado.

Errores

- `EntityNotFoundException`
- `OperationTimeoutException`

Acción GetClassifiers (Python: get_classifiers)

Muestra todos los objetos de tipo clasificador del catálogo de datos.

Solicitud

- `MaxResults`: número (entero) que no es inferior a 1 ni es superior a 1000.

  Tamaño de la lista que se devolverá (opcional).
- `NextToken`: cadena UTF-8.

  Token de continuación opcional.

Respuesta

- `Classifiers`: matriz de objetos Clasificador.

  La lista de objetos de tipo clasificador solicitada.
- `NextToken`: cadena UTF-8.

  Token de continuación.

Errores

- `OperationTimeoutException`

Acción UpdateClassifier (Python: update_classifier)

Modifica un clasificador existente (`GrokClassifier`, `XMLClassifier`, `JsonClassifier` o `CsvClassifier`, en función del campo que esté presente).

Solicitud

- `GrokClassifier`: un objeto UpdateGrokClassifierRequest.

Objeto `GrokClassifier` con campos actualizados.

- `XMLClassifier`: un objeto UpdateXMLClassifierRequest.

  Objeto `XMLClassifier` con campos actualizados.

- `JsonClassifier`: un objeto UpdateJsonClassifierRequest.

  Objeto `JsonClassifier` con campos actualizados.

- `CsvClassifier`: un objeto UpdateCsvClassifierRequest.

  Objeto `CsvClassifier` con campos actualizados.

Respuesta

- Sin parámetros de respuesta.

Errores

- `InvalidInputException`
- `VersionMismatchException`
- `EntityNotFoundException`
- `OperationTimeoutException`

La API del rastreador

La API de rastreadores describe los tipos de datos de los AWS Glue rastreadores, junto con la API para crear, eliminar, actualizar y enumerar los rastreadores.

Tipos de datos

- Estructura de rastreador
- Estructura de programación
- CrawlerTargets estructura
- Estructura S3Target
- DeltaCatalogTarget Estructura S3

- DeltaDirectTarget Estructura S3
- JdbcTarget estructura
- Estructura MongoDBTarget
- Estructura DynamoDBTarget
- DeltaTarget estructura
- IcebergTarget estructura
- HudiTarget estructura
- CatalogTarget estructura
- CrawlerMetrics estructura
- CrawlerHistory estructura
- CrawlsFilter estructura
- SchemaChangePolicy estructura
- LastCrawlInfo estructura
- RecrawlPolicy estructura
- LineageConfiguration estructura
- LakeFormationConfiguration estructura

Estructura de rastreador

Especifica un programa de rastreador que examina un origen de datos y utiliza clasificadores para intentar determinar su esquema. Si ejecuta la operación correctamente, el rastreador registra los metadatos relativos al origen de los datos en AWS Glue Data Catalog.

Campos

- `Name`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre del rastreador.

- `Role`: cadena UTF-8.

  El nombre de recurso de Amazon (ARN) de un rol de IAM que se utiliza para obtener acceso a los recursos del cliente como, por ejemplo, los datos de Amazon Simple Storage Service (Amazon S3).

- `Targets`: un objeto CrawlerTargets.

  Colección de objetivos del rastreo.

- `DatabaseName`: cadena UTF-8.

  El nombre de la base de datos donde se almacena la salida del rastreador.

- `Description`: cadena de descripción de un máximo de 2048 bytes de largo, que coincide con el URI address multi-line string pattern.

  Descripción del rastreador.

- `Classifiers`: matriz de cadenas UTF-8.

  Una lista de cadenas UTF-8 que especifican los clasificadores personalizados que están asociados al rastreador.

- `RecrawlPolicy`: un objeto RecrawlPolicy.

  Política que especifica si se debe rastrear de nuevo todo el conjunto de datos o si se deben rastrear sólo las carpetas que se agregaron desde la última ejecución del rastreador.

- `SchemaChangePolicy`: un objeto SchemaChangePolicy.

  La política que especifica los comportamientos de actualización y eliminación del rastreador.

- `LineageConfiguration`: un objeto LineageConfiguration.

  Configuración que especifica si el linaje de datos está habilitado para el rastreador.

- `State`: cadena UTF-8 (valores válidos: `READY` | `RUNNING` | `STOPPING`).

  Indica si el rastreador se está ejecutando o si queda pendiente una ejecución.

- `TablePrefix`: cadena UTF-8 de 128 bytes de largo como máximo.

  Prefijo añadido a los nombres de tablas que se crean.

- `Schedule`: un objeto Programación.

  En rastreadores programados, el programa cuando se ejecuta el rastreador.

- `CrawlElapsedTime`: número (largo).

  Si el rastreador se está ejecutando, contiene el tiempo transcurrido total desde que comenzó el último rastreo.

- `CreationTime`: marca temporal.

La hora de creación del rastreador.

- `LastUpdated`: marca temporal.

  La hora de la última actualización del rastreador.

- `LastCrawl`: un objeto LastCrawlInfo.

  Estado del último rastreo e información de error potencial si se produjo un error.

- `Version`: número (largo).

  Versión del rastreador.

- `Configuration`: cadena UTF-8.

  Información de configuración del rastreador. Esta cadena JSON con varias versiones permite a los usuarios especificar aspectos del comportamiento de un rastreador. Para obtener más información, consulte Establecimiento de opciones de configuración de rastreadores.

- `CrawlerSecurityConfiguration`: cadena UTF-8 de 128 bytes de largo como máximo.

  El nombre de la estructura `SecurityConfiguration` que va a utilizar este rastreador.

- `LakeFormationConfiguration`: un objeto LakeFormationConfiguration.

  Especifica si el rastreador debe usar AWS Lake Formation credenciales para el rastreador en lugar de las credenciales del rol de IAM.

Estructura de programación

Objeto de programación que usa una instrucción `cron` para programar un evento.

Campos

- `ScheduleExpression`: cadena UTF-8.

  Expresión `cron` utilizada para especificar el programa (consulte Programaciones basadas en tiempo para trabajos y rastreadores). Por ejemplo, para ejecutar algo todos los días a las 12:15 UTC, especifique: `cron(15 12 * * ? *)`.

- `State`: cadena UTF-8 (valores válidos: `SCHEDULED` | `NOT_SCHEDULED` | `TRANSITIONING`).

  Estado del programa.

CrawlerTargets estructura

Especifica los almacenes de datos que deben rastrearse.

Campos

- `S3Targets`: matriz de objetos S3Target.

  Especifica los destinos de Amazon Simple Storage Service (Amazon S3).
- `JdbcTargets`: matriz de objetos JdbcTarget.

  Especifica los destinos de JDBC.
- `MongoDBTargets`: matriz de objetos MongoDBTarget.

  Especifica los orígenes de Amazon DocumentDB o MongoDB.
- `DynamoDBTargets`: matriz de objetos DynamoDBTarget.

  Especifica los destinos de Amazon DynamoDB.
- `CatalogTargets`: matriz de objetos CatalogTarget.

  Especifica AWS Glue Data Catalog los objetivos.
- `DeltaTargets`: matriz de objetos DeltaTarget.

  Especifica los destinos de almacenamiento de datos Delta.
- `IcebergTargets`: matriz de objetos IcebergTarget.

  Especifica los destinos de almacenamiento de datos Apache Iceberg.
- `HudiTargets`: matriz de objetos HudiTarget.

  Especifica los destinos de almacenamiento de datos Apache Hudi.

Estructura S3Target

Especifica un almacén de datos en Amazon Simple Storage Service (Amazon S3).

Campos

- `Path`: cadena UTF-8.

  Ruta al destino de Amazon S3.

- `Exclusions`: matriz de cadenas UTF-8.

  Lista de patrones glob utilizados para excluir elementos del rastreo. Para obtener más información, consulte la sección acerca de cómo Catalogar tablas con un rastreador.

- `ConnectionName`: cadena UTF-8.

  Nombre de una conexión que permite a un trabajo o rastreador acceder a los datos de Amazon S3 dentro de un entorno de Amazon Virtual Private Cloud (Amazon VPC).

- `SampleSize`: número (entero).

  Establece el número de archivos de cada carpeta que se van a rastrear al rastrear archivos de ejemplo en un conjunto de datos. Si no se establece, se rastrean todos los archivos. Un valor válido es un entero entre 1 y 249.

- `EventQueueArn`: cadena UTF-8.

  Un ARN de Amazon SQS válido. Por ejemplo, `arn:aws:sqs:region:account:sqs`.

- `DlqEventQueueArn`: cadena UTF-8.

  Un ARN de Amazon SQS válido con mensajes fallidos. Por ejemplo, `arn:aws:sqs:region:account:deadLetterQueue`.

DeltaCatalogTarget Estructura S3

Especifica un destino que escribe en una fuente de datos de Delta Lake del catálogo AWS Glue de datos.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del destino de datos.

- `Inputs`: Obligatorio: matriz de cadenas UTF-8, con una cadena como mínimo y una cadena como máximo.

  Los nodos que son entradas para el destino de datos.

- `PartitionKeys`: matriz de cadenas UTF-8.

  Especifica la partición nativa mediante una secuencia de claves.

- `Table`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de la tabla de la base de datos de escritura.

- `Database`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de la base de datos de escritura.

- `AdditionalOptions`: matriz de mapas de pares clave-valor.

  Cada clave es una cadena UTF-8 que coincide con el Custom string pattern #40.

  Cada valor es una cadena UTF-8 que coincide con el Custom string pattern #40.

  Especifica opciones de conexión adicionales para el conector.

- `SchemaChangePolicy`: un objeto CatalogSchemaChangePolicy.

  Una política que especifica los comportamientos de actualización del rastreador.

DeltaDirectTarget Estructura S3

Especifica un destino que escribe en una fuente de datos de Delta Lake en Amazon S3.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del destino de datos.

- `Inputs`: Obligatorio: matriz de cadenas UTF-8, con una cadena como mínimo y una cadena como máximo.

  Los nodos que son entradas para el destino de datos.

- `PartitionKeys`: matriz de cadenas UTF-8.

  Especifica la partición nativa mediante una secuencia de claves.

- `Path`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  La ruta de Amazon S3 del origen de datos de Delta Lake en la que desea escribir.

- `Compression`: obligatorio: cadena UTF-8 (valores válidos: `uncompressed="UNCOMPRESSED"` | `snappy="SNAPPY"`).

Especifica la forma en que los datos se comprimen. Po lo general no es necesario si los datos tienen una extensión de archivo estándar. Los posibles valores son `"gzip"` y `"bzip"`).

- `Format` – Obligatorio: cadena UTF-8 (valores válidos: `json="JSON"` | `csv="CSV"` | `avro="AVRO"` | `orc="ORC"` | `parquet="PARQUET"` | `hudi="HUDI"` | `delta="DELTA"`).

  Especifica el formato de salida de datos para el destino.

- `AdditionalOptions`: matriz de mapas de pares clave-valor.

  Cada clave es una cadena UTF-8 que coincide con el Custom string pattern #40.

  Cada valor es una cadena UTF-8 que coincide con el Custom string pattern #40.

  Especifica opciones de conexión adicionales para el conector.

- `SchemaChangePolicy`: un objeto DirectSchemaChangePolicy.

  Una política que especifica los comportamientos de actualización del rastreador.

JdbcTarget estructura

Especifica un almacén de datos JDBC donde efectuar el rastreo.

Campos

- `ConnectionName`: cadena UTF-8.

  Nombre de la conexión que se utilizará para establecer conexión con el destino de JDBC.

- `Path`: cadena UTF-8.

  Ruta del destino de JDBC.

- `Exclusions`: matriz de cadenas UTF-8.

  Lista de patrones glob utilizados para excluir elementos del rastreo. Para obtener más información, consulte la sección acerca de cómo Catalogar tablas con un rastreador.

- `EnableAdditionalMetadata`: matriz de cadenas UTF-8.

  Especifique un valor de `RAWTYPES` o `COMMENTS` para habilitar metadatos adicionales en las respuestas de la tabla. `RAWTYPES` proporciona el tipo de datos de nivel nativo. `COMMENTS` proporciona comentarios asociados a una columna o tabla de la base de datos.

Si no necesita metadatos adicionales, mantenga el campo vacío.

Estructura MongoDBTarget

Especifica un almacén de datos Amazon DocumentDB o MongoDB donde realizar el rastreo.

Campos

- `ConnectionName`: cadena UTF-8.

  Nombre de la conexión que se utilizará para establecer conexión con el origen de Amazon DocumentDB o MongoDB.

- `Path`: cadena UTF-8.

  Ruta de acceso del origen de Amazon DocumentDB o MongoDB (base de datos/recopilación).

- `ScanAll`: booleano.

  Indica si se deben analizar todos los registros o si se deben muestrear filas de la tabla. Escanear todos los registros puede tardar mucho tiempo cuando la tabla no es una tabla de alto rendimiento.

  Un valor de `true` significa que hay que escanear todos los registros, mientras que un valor de `false` significa que se deben muestrear los registros. Si no se especifica ningún valor, el valor predeterminado es `true`.

Estructura DynamoDBTarget

Especifica una tabla de Amazon DynamoDB para rastrear.

Campos

- `Path`: cadena UTF-8.

  Nombre de la tabla de DynamoDB donde efectuar el rastreo.

- `scanAll`: booleano.

  Indica si se deben analizar todos los registros o si se deben muestrear filas de la tabla. Escanear todos los registros puede tardar mucho tiempo cuando la tabla no es una tabla de alto rendimiento.

Un valor de `true` significa que hay que escanear todos los registros, mientras que un valor de `false` significa que se deben muestrear los registros. Si no se especifica ningún valor, el valor predeterminado es `true`.

- `scanRate`: número (doble).

  El porcentaje de unidades de capacidad de lectura configuradas que utilizará el AWS Glue rastreador. Unidades de capacidad de lectura es un término definido por DynamoDB y es un valor numérico que actúa como limitador de velocidad del número de lecturas que se pueden realizar en esa tabla por segundo.

  Los valores válidos son nulos o un valor entre 0,1 y 1,5. Se utiliza un valor nulo cuando el usuario no proporciona un valor y el valor predeterminado es 0,5 de la unidad de capacidad de lectura configurada (en tablas aprovisionadas) o 0,25 de la unidad de capacidad de lectura máxima configurada (en tablas que utilizan el modo bajo demanda).

DeltaTarget estructura

Especifica un almacenamiento de datos Delta para rastrear una o más tablas Delta.

Campos

- `DeltaTables`: matriz de cadenas UTF-8.

  Una lista de las rutas de Simple Storage Service (Amazon S3) hacia las tablas Delta.

- `ConnectionName`: cadena UTF-8.

  Nombre de la conexión que se utilizará para establecer conexión con el destino de la tabla Delta.

- `WriteManifest`: booleano.

  Especifica si se deben escribir los archivos de manifiesto en la ruta de la tabla Delta.

- `CreateNativeDeltaTable`: booleano.

  Especifica si el rastreador creará tablas nativas para permitir la integración con los motores de consulta que permiten consultar directamente el registro de transacciones de Delta.

IcebergTarget estructura

Especifica un origen de datos de Apache Iceberg en la que se almacenan las tablas de Iceberg en Amazon S3.

Campos

- `Paths`: matriz de cadenas UTF-8.

  Una o más Amazon S3 rutas que contienen carpetas de metadatos de Iceberg como`s3://bucket/prefix`.

- `ConnectionName`: cadena UTF-8.

  Nombre de la conexión que se utilizará para establecer conexión con el destino de Iceberg.

- `Exclusions`: matriz de cadenas UTF-8.

  Lista de patrones glob utilizados para excluir elementos del rastreo. Para obtener más información, consulte la sección acerca de cómo Catalogar tablas con un rastreador.

- `MaximumTraversalDepth`: número (entero).

  La profundidad máxima de las Amazon S3 rutas que el rastreador puede recorrer para descubrir la carpeta de metadatos de Iceberg en su ruta. Amazon S3 Se utiliza para limitar el tiempo de ejecución del rastreador.

HudiTarget estructura

Especifica un origen de datos Apache Hudi.

Campos

- `Paths`: matriz de cadenas UTF-8.

  Matriz de cadenas de Amazon S3 ubicación para Hudi, cada una de las cuales indica la carpeta raíz en la que residen los archivos de metadatos de una tabla Hudi. La carpeta Hudi puede estar ubicada en una carpeta secundaria de la carpeta raíz.

  El rastreador escaneará todas las carpetas situadas debajo de una ruta para una carpeta Hudi.

- `ConnectionName`: cadena UTF-8.

Nombre de la conexión que se utilizará para establecer conexión con el destino de Hudi. Si sus archivos Hudi están almacenados en buckets que requieren autorización de VPC, puede configurar sus propiedades de conexión aquí.

- `Exclusions`: matriz de cadenas UTF-8.

  Lista de patrones glob utilizados para excluir elementos del rastreo. Para obtener más información, consulte la sección acerca de cómo Catalogar tablas con un rastreador.

- `MaximumTraversalDepth`: número (entero).

  La profundidad máxima de las Amazon S3 rutas que el rastreador puede recorrer para descubrir la carpeta de metadatos de Hudi en su ruta. Amazon S3 Se utiliza para limitar el tiempo de ejecución del rastreador.

CatalogTarget estructura

Especifica un AWS Glue Data Catalog objetivo.

Campos

- `DatabaseName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre de la base de datos que se va a sincronizar.

- `Tables`: obligatorio: matriz de cadenas UTF-8, con una cadena como mínimo.

  Lista de las tablas que se van a sincronizar.

- `ConnectionName`: cadena UTF-8.

  El nombre de la conexión de una tabla del Catálogo de datos respaldada por Amazon S3 que se rastreará al utilizar un tipo de conexión `Catalog` emparejado con un tipo de conexión `NETWORK`.

- `EventQueueArn`: cadena UTF-8.

  Un ARN de Amazon SQS válido. Por ejemplo, `arn:aws:sqs:region:account:sqs`.

- `DlqEventQueueArn`: cadena UTF-8.

  Un ARN de Amazon SQS válido con mensajes fallidos. Por ejemplo, `arn:aws:sqs:region:account:deadLetterQueue`.

CrawlerMetrics estructura

Métricas de un rastreador especificado.

Campos

- `CrawlerName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre del rastreador.
- `TimeLeftSeconds`: número (doble), cero como máximo.

  Tiempo estimado que queda para completar un rastreo que se está ejecutando.
- `StillEstimating`: booleano.

  True si el rastreador sigue calculando cuánto tiempo se tardará en completar esta ejecución.
- `LastRuntimeSeconds`: número (doble), cero como máximo.

  Duración de la ejecución más reciente del rastreador indicada en segundos.
- `MedianRuntimeSeconds`: número (doble), cero como máximo.

  Duración media de las ejecuciones de este rastreador indicada en segundos.
- `TablesCreated`: número (entero), cero como máximo.

  Número de tablas creadas por este rastreador.
- `TablesUpdated`: número (entero), cero como máximo.

  Número de tablas actualizadas por este rastreador.
- `TablesDeleted`: número (entero), cero como máximo.

  Número de tablas eliminadas por este rastreador.

CrawlerHistory estructura

Contiene la información de la ejecución de un rastreador.

Campos

- `CrawlId`: cadena UTF-8.

Identificador UUID de cada rastreo.

- `State`: cadena UTF-8 (valores válidos: `RUNNING` | `COMPLETED` | `FAILED` | `STOPPED`).

  El estado del rastreo.

- `StartTime`: marca temporal.

  La fecha y hora en las que se inició el rastreo.

- `EndTime`: marca temporal.

  La fecha y hora en las que terminó el rastreo.

- `Summary`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Un resumen de la ejecución del rastreo específico en JSON. Contiene las tablas y particiones del catálogo que se agregaron, actualizaron o eliminaron.

- `ErrorMessage`: cadena de descripción de un máximo de 2048 bytes de largo, que coincide con el URI address multi-line string pattern.

  Si se produjo un error, el mensaje de error asociado con el rastreo.

- `LogGroup`: cadena UTF-8, con 1 byte de largo como mínimo y 512 bytes de largo como máximo, que coincide con el Log group string pattern.

  El grupo de registros asociado al rastreo.

- `LogStream`: cadena UTF-8, con 1 byte de largo como mínimo y 512 bytes de largo como máximo, que coincide con el Log-stream string pattern.

  El flujo de registros asociado al rastreo.

- `MessagePrefix`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El prefijo de un CloudWatch mensaje sobre este rastreo.

- `DPUHour`: número (doble), cero como máximo.

  El número de unidades de procesamiento de datos (DPU) utilizadas en horas para el rastreo.

CrawlsFilter estructura

Una lista de campos, comparadores y valores que se pueden utilizar para filtrar las ejecuciones de rastreador de un rastreador específico.

Campos

- `FieldName`: cadena UTF-8 (valores válidos: `CRAWL_ID` | `STATE` | `START_TIME` | `END_TIME` | `DPU_HOUR`).

  Una clave que se utiliza para filtrar las ejecuciones del rastreador para un rastreador específico. Los valores válidos para cada uno de los nombres de campo son:
  - `CRAWL_ID`: cadena que representa el identificador de UUID de un rastreo.
  - `STATE`: una cadena de caracteres que representa el estado del rastreo.
  - `START_TIME` y `END_TIME`: la marca de tiempo Epoch en milisegundos.
  - `DPU_HOUR`: el número de horas de unidad de procesamiento de datos (DPU) utilizadas para el rastreo.
- `FilterOperator`: cadena UTF-8 (valores válidos: `GT` | `GE` | `LT` | `LE` | `EQ` | `NE`).

  Un comparador definido que opera con el valor. Los operadores disponibles son:
  - `GT`: mayor que.
  - `GE`: mayor o igual que.
  - `LT`: menor que.
  - `LE`: menor o igual que.
  - `EQ`: igual que.
  - `NE`: no es igual que.
- `FieldValue`: cadena UTF-8.

  El valor proporcionado para la comparación en el campo de rastreo.

SchemaChangePolicy estructura

Una política que especifica los comportamientos de actualización y eliminación del rastreador.

Campos

- `UpdateBehavior`: cadena UTF-8 (valores válidos: `LOG` | `UPDATE_IN_DATABASE`).

Comportamiento de actualización cuando el rastreador encuentra un esquema cambiado.

- `DeleteBehavior`: cadena UTF-8 (valores válidos: `LOG` | `DELETE_FROM_DATABASE` | `DEPRECATE_IN_DATABASE`).

  Comportamiento de eliminación cuando el rastreador encuentra un objeto eliminado.

LastCrawlInfo estructura

Información de estado y de error sobre el rastreo más reciente.

Campos

- `Status`: cadena UTF-8 (valores válidos: `SUCCEEDED` | `CANCELLED` | `FAILED`).

  Estado del último rastreo.

- `ErrorMessage`: cadena de descripción de un máximo de 2048 bytes de largo, que coincide con el URI address multi-line string pattern.

  Si se produjo un error, la información de error sobre el último rastreo.

- `LogGroup`: cadena UTF-8, con 1 byte de largo como mínimo y 512 bytes de largo como máximo, que coincide con el Log group string pattern.

  Grupo de registros del último rastreo.

- `LogStream`: cadena UTF-8, con 1 byte de largo como mínimo y 512 bytes de largo como máximo, que coincide con el Log-stream string pattern.

  Flujo de registros del último rastreo.

- `MessagePrefix`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Prefijo de un mensaje sobre este rastreo.

- `StartTime`: marca temporal.

  Hora en la que se inició el rastreo.

RecrawlPolicy estructura

Al rastrear un origen de datos de Amazon S3 después de completar el primer rastreo, especifica si se debe rastrear de nuevo todo el conjunto de datos o si se deben rastrear sólo las carpetas que se agregaron desde la última ejecución del rastreador. Para obtener más información, consulte Rastreo progresivo en AWS Glue en la guía para desarrolladores.

Campos

- `RecrawlBehavior`: cadena UTF-8 (valores válidos: `CRAWL_EVERYTHING` | `CRAWL_NEW_FOLDERS_ONLY` | `CRAWL_EVENT_MODE`).

  Especifica si se debe rastrear de nuevo todo el conjunto de datos o si se deben rastrear sólo las carpetas que se agregaron desde la última ejecución del rastreador.

  Un valor de `CRAWL_EVERYTHING` especifica volver a rastrear todo el conjunto de datos.

  Un valor de `CRAWL_NEW_FOLDERS_ONLY` especifica el rastreo sólo de carpetas que se agregaron desde la última ejecución del rastreador.

  Un valor de `CRAWL_EVENT_MODE` especifica rastrear solo los cambios identificados por los eventos de Amazon S3.

LineageConfiguration estructura

Especifica los parámetros de configuración del linaje de datos para el rastreador.

Campos

- `CrawlerLineageSettings`: cadena UTF-8 (valores válidos: `ENABLE` | `DISABLE`).

  Especifica si el linaje de datos está habilitado para el rastreador. Los valores válidos son:

  - HABILITAR: habilita el linaje de datos para el rastreador
  - DESHABILITAR: deshabilita el linaje de datos para el rastreador

LakeFormationConfiguration estructura

Especifica los parámetros de AWS Lake Formation configuración del rastreador.

Campos

- `UseLakeFormationCredentials`: booleano.

  Especifica si se van a utilizar AWS Lake Formation las credenciales del rastreador en lugar de las credenciales del rol de IAM.
- `AccountId`: cadena UTF-8, no más de 12 bytes de largo.

  Necesaria para rastreos de cuentas cruzadas. Para los mismos rastreos de cuentas que los datos de destino, esto se puede dejar como nulo.

Operaciones

- CreateCrawler acción (Python: create_crawler)
- DeleteCrawler acción (Python: delete_crawler)
- GetCrawler acción (Python: get_crawler)
- GetCrawlers acción (Python: get_crawlers)
- GetCrawlerMetrics acción (Python: get_crawler_metrics)
- UpdateCrawler acción (Python: update_crawler)
- StartCrawler acción (Python: start_crawler)
- StopCrawler acción (Python: stop_crawler)
- BatchGetCrawlers acción (Python: batch_get_crawlers)
- ListCrawlers acción (Python: list_crawlers)
- ListCrawls acción (Python: list_crawls)

CreateCrawler acción (Python: create_crawler)

Crea un nuevo rastreador con objetivos, rol, configuración y programación opcional especificados. Se debe especificar al menos un objetivo de rastreo, en el campo `s3Targets`, el campo `jdbcTargets` o el campo `DynamoDBTargets`.

Solicitud

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

Nombre del rastreador nuevo.

- `Role` – Obligatorio: cadena UTF-8.

  El rol de IAM o nombre de recursos de Amazon (ARN) de un rol de IAM que el nuevo rastreador utiliza para obtener acceso a recursos de los clientes.

- `DatabaseName`: cadena UTF-8.

  La AWS Glue base de datos en la que se escriben los resultados, como:. `arn:aws:daylight:us-east-1::database/sometable/*`

- `Description`: cadena de descripción de un máximo de 2048 bytes de largo, que coincide con el URI address multi-line string pattern.

  Descripción del nuevo rastreador.

- `Targets`: obligatorio: objeto CrawlerTargets.

  Lista de colecciones de objetivos donde realizar el rastreo.

- `Schedule`: cadena UTF-8.

  Expresión `cron` utilizada para especificar el programa (consulte Programaciones basadas en tiempo para trabajos y rastreadores). Por ejemplo, para ejecutar algo todos los días a las 12:15 UTC, especifique: `cron(15 12 * * ? *)`.

- `Classifiers`: matriz de cadenas UTF-8.

  Lista de clasificadores personalizados que el usuario ha registrado. De forma predeterminada, todos los clasificadores integrados se incluyen en un rastreo, pero estos clasificadores personalizados siempre anulan los clasificadores predeterminados de una determinada clasificación.

- `TablePrefix`: cadena UTF-8 de 128 bytes de largo como máximo.

  Prefijo de tabla utilizado para las tablas de catálogo que se crean.

- `SchemaChangePolicy`: un objeto SchemaChangePolicy.

  La política para el comportamiento de actualización y eliminación del rastreador.

- `RecrawlPolicy`: un objeto RecrawlPolicy.

  Política que especifica si se debe rastrear de nuevo todo el conjunto de datos o si se deben rastrear sólo las carpetas que se agregaron desde la última ejecución del rastreador.

- `LineageConfiguration`: un objeto LineageConfiguration.

  Especifica los parámetros de configuración del linaje de datos para el rastreador.

- `LakeFormationConfiguration`: un objeto LakeFormationConfiguration.

  Especifica los ajustes de AWS Lake Formation configuración del rastreador.

- `Configuration`: cadena UTF-8.

  Información de configuración del rastreador. Esta cadena JSON con varias versiones permite a los usuarios especificar aspectos del comportamiento de un rastreador. Para obtener más información, consulte Establecimiento de opciones de configuración de rastreadores.

- `CrawlerSecurityConfiguration`: cadena UTF-8 de 128 bytes de largo como máximo.

  El nombre de la estructura `SecurityConfiguration` que va a utilizar este rastreador.

- `Tags`: matriz de mapas de pares de clave-valor, con 50 pares como máximo.

  Cada clave es una cadena UTF-8 con una longitud de entre 1 y 128 bytes.

  Cada valor es una cadena UTF-8 que no tiene más de 256 bytes de largo.

  Etiquetas que se van a utilizar con esta solicitud de rastreador. Puede utilizar etiquetas para limitar el acceso al rastreador. Para obtener más información sobre las etiquetas AWS Glue, consulte AWS Etiquetas incluidas AWS Glue en la guía para desarrolladores.

Respuesta

- Sin parámetros de respuesta.

Errores

- `InvalidInputException`
- `AlreadyExistsException`
- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`

DeleteCrawler acción (Python: delete_crawler)

Elimina un rastreador específico del AWS Glue Data Catalog, a menos que el estado del rastreador sea. `RUNNING`

Solicitud

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre del rastreador que se eliminará.

Respuesta

- Sin parámetros de respuesta.

Errores

- `EntityNotFoundException`
- `CrawlerRunningException`
- `SchedulerTransitioningException`
- `OperationTimeoutException`

GetCrawler acción (Python: get_crawler)

Recupera metadatos para un rastreador especificado.

Solicitud

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre del rastreador para el que se recuperarán los metadatos.

Respuesta

- `Crawler`: un objeto Rastreador.

  Los metadatos para el rastreador especificado.

Errores

- `EntityNotFoundException`
- `OperationTimeoutException`

GetCrawlers acción (Python: get_crawlers)

Recupera metadatos para todos los rastreadores definidos en la cuenta del cliente.

Solicitud

- `MaxResults`: número (entero) que no es inferior a 1 ni es superior a 1000.

  Número de rastreadores que se devolverá en cada llamada.
- `NextToken`: cadena UTF-8.

  Token de continuación, si se trata de una solicitud de continuidad.

Respuesta

- `Crawlers`: matriz de objetos Rastreador.

  Lista de metadatos de rastreador.
- `NextToken`: cadena UTF-8.

  Token de continuación, si la lista devuelta no ha llegado al final de los metadatos definidos en esta cuenta de cliente.

Errores

- `OperationTimeoutException`

GetCrawlerMetrics acción (Python: get_crawler_metrics)

Recupera métricas sobre rastreadores especificados.

Solicitud

- `CrawlerNameList`: matriz de cadenas UTF-8, con 100 cadenas como máximo.

Lista de los nombres de rastreadores sobre los que se recuperarán métricas.

- `MaxResults`: número (entero) que no es inferior a 1 ni es superior a 1000.

  Tamaño máximo de una lista que se devolverá.

- `NextToken`: cadena UTF-8.

  Token de continuación si se trata de una llamada de continuidad.

Respuesta

- `CrawlerMetricsList`: matriz de objetos CrawlerMetrics.

  Lista de métricas para el rastreador especificado.

- `NextToken`: cadena UTF-8.

  Token de continuación, si la lista devuelta no contiene la última métrica disponible.

Errores

- `OperationTimeoutException`

UpdateCrawler acción (Python: update_crawler)

Actualiza un rastreador. Si un rastreador se está ejecutando, debe detenerlo utilizando `StopCrawler` antes de actualizarlo.

Solicitud

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre del rastreador nuevo.

- `Role`: cadena UTF-8.

  El rol de IAM o nombre de recurso de Amazon (ARN) de un rol de IAM que utiliza el nuevo rastreador para acceder a los recursos del cliente.

- `DatabaseName`: cadena UTF-8.

La AWS Glue base de datos donde se almacenan los resultados, como:. `arn:aws:daylight:us-east-1::database/sometable/*`

- `Description`: cadena UTF-8 con un máximo de 2048 bytes de largo, que coincide con el URI address multi-line string pattern.

  Descripción del nuevo rastreador.

- `Targets`: un objeto CrawlerTargets.

  Lista de objetivos del rastreo.

- `Schedule`: cadena UTF-8.

  Expresión `cron` utilizada para especificar el programa (consulte Programaciones basadas en tiempo para trabajos y rastreadores). Por ejemplo, para ejecutar algo todos los días a las 12:15 UTC, especifique: `cron(15 12 * * ? *)`.

- `Classifiers`: matriz de cadenas UTF-8.

  Lista de clasificadores personalizados que el usuario ha registrado. De forma predeterminada, todos los clasificadores integrados se incluyen en un rastreo, pero estos clasificadores personalizados siempre anulan los clasificadores predeterminados de una determinada clasificación.

- `TablePrefix`: cadena UTF-8 de 128 bytes de largo como máximo.

  Prefijo de tabla utilizado para las tablas de catálogo que se crean.

- `SchemaChangePolicy`: un objeto SchemaChangePolicy.

  La política para el comportamiento de actualización y eliminación del rastreador.

- `RecrawlPolicy`: un objeto RecrawlPolicy.

  Política que especifica si se debe rastrear de nuevo todo el conjunto de datos o si se deben rastrear sólo las carpetas que se agregaron desde la última ejecución del rastreador.

- `LineageConfiguration`: un objeto LineageConfiguration.

  Especifica los parámetros de configuración del linaje de datos para el rastreador.

- `LakeFormationConfiguration`: un objeto LakeFormationConfiguration.

  Especifica los ajustes de AWS Lake Formation configuración del rastreador.

- `Configuration`: cadena UTF-8.

Información de configuración del rastreador. Esta cadena JSON con varias versiones permite a los usuarios especificar aspectos del comportamiento de un rastreador. Para obtener más información, consulte Establecimiento de opciones de configuración de rastreadores.

- `CrawlerSecurityConfiguration`: cadena UTF-8 de 128 bytes de largo como máximo.

  El nombre de la estructura `SecurityConfiguration` que va a utilizar este rastreador.

Respuesta

- Sin parámetros de respuesta.

Errores

- `InvalidInputException`
- `VersionMismatchException`
- `EntityNotFoundException`
- `CrawlerRunningException`
- `OperationTimeoutException`

StartCrawler acción (Python: start_crawler)

Comienza un rastreo utilizando el rastreo especificado, independientemente de lo que esté programado. Si el rastreador ya se está ejecutando, devuelve un. CrawlerRunningException

Solicitud

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre del rastreador que se iniciará.

Respuesta

- Sin parámetros de respuesta.

Errores

- `EntityNotFoundException`
- `CrawlerRunningException`
- `OperationTimeoutException`

StopCrawler acción (Python: stop_crawler)

Si el rastreador especificado está en ejecución, se detiene el rastreo.

Solicitud

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre del rastreador que se detendrá.

Respuesta

- Sin parámetros de respuesta.

Errores

- `EntityNotFoundException`
- `CrawlerNotRunningException`
- `CrawlerStoppingException`
- `OperationTimeoutException`

BatchGetCrawlers acción (Python: batch_get_crawlers)

Devuelve la lista de metadatos de recursos de una determinada lista de nombres de rastreadores. Después de llamar a la operación `ListCrawlers`, puede llamar a esta operación para obtener acceso a los datos a los que ha concedido permisos. Esta operación admite todos los permisos de IAM, incluidas las condiciones de permisos que utilizan etiquetas.

Solicitud

- `CrawlerNames` – Obligatorio: matriz de cadenas UTF-8, con 100 cadenas como máximo.

Una lista de nombres de rastreadores, que pueden ser los nombres devueltos en la operación `ListCrawlers`.

Respuesta

- `Crawlers`: matriz de objetos Rastreador.

  Lista de definiciones de rastreadores.
- `CrawlersNotFound`: matriz de cadenas UTF-8, con 100 cadenas como máximo.

  Una lista de nombres de rastreadores que no se han encontrado.

Errores

- `InvalidInputException`
- `OperationTimeoutException`

ListCrawlers acción (Python: list_crawlers)

Recupera los nombres de todos los recursos del rastreador de esta AWS cuenta o los recursos con la etiqueta especificada. Esta operación permite ver qué recursos están disponibles en la cuenta y sus nombres.

Esta operación toma el campo `Tags` opcional, que se puede utilizar como filtro en la respuesta para que los recursos etiquetados se devuelvan agrupados. Si decide utilizar el filtrado de etiquetas, solo se devolverán los recursos con la etiqueta especificada.

Solicitud

- `MaxResults`: número (entero) que no es inferior a 1 ni es superior a 1000.

  Tamaño máximo de una lista que se devolverá.
- `NextToken`: cadena UTF-8.

  Token de continuación, si se trata de una solicitud de continuidad.
- `Tags`: matriz de mapas de pares de clave-valor, con 50 pares como máximo.

  Cada clave es una cadena UTF-8 con una longitud de entre 1 y 128 bytes.

Cada valor es una cadena UTF-8 que no tiene más de 256 bytes de largo.

Especifica que se devuelvan solamente los recursos etiquetados.

Respuesta

- `CrawlerNames`: matriz de cadenas UTF-8, con 100 cadenas como máximo.

  Nombres de todos los rastreadores de la cuenta o de los rastreadores con las etiquetas especificadas.
- `NextToken`: cadena UTF-8.

  Token de continuación, si la lista devuelta no contiene la última métrica disponible.

Errores

- `OperationTimeoutException`

ListCrawls acción (Python: list_crawls)

Devuelve todos los rastreos de un rastreador especificado. Devuelve solo los rastreos que se han producido desde la fecha de lanzamiento de la función de historial del rastreador y solo conserva hasta 12 meses de rastreo. No se devolverán los rastreos más antiguos.

Puede utilizar esta API para:

- Recupera todos los rastreos de un rastreador especificado.
- Recupera todos los rastreos de un rastreador específico dentro de un recuento limitado.
- Recupera todos los rastreos de un rastreador específico en un rango de tiempo específico.
- Recupera todos los rastreos de un rastreador especificado con un estado, un ID de rastreo o un valor de hora de DPU en particular.

Solicitud

- `CrawlerName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

El nombre del rastreador cuyo valor desea recuperar.

- `MaxResults`: número (entero) que no es inferior a 1 ni es superior a 1000.

  El número máximo de resultados que devolver. El valor predeterminado es 20 y el máximo es 100.

- `Filters`: matriz de objetos CrawlsFilter.

  Filtra los rastreos de acuerdo con los criterios especificados en una lista de objetos de `CrawlsFilter`.

- `NextToken`: cadena UTF-8.

  Token de continuación si se trata de una llamada de continuidad.

Respuesta

- `Crawls`: matriz de objetos CrawlerHistory.

  Una lista de objetos de `CrawlerHistory` que representan las ejecuciones de rastreo que cumplen sus criterios.

- `NextToken`: cadena UTF-8.

  Token de continuación para paginar la lista de tokens obtenida; se devuelve si el segmento actual de la lista no es el último.

Errores

- `EntityNotFoundException`
- `OperationTimeoutException`
- `InvalidInputException`

API de estadísticas de columna

La API de estadísticas de columnas describe las API de AWS Glue para devolver las estadísticas de las columnas de una tabla.

Tipos de datos

- Estructura ColumnStatisticsTaskRun

- Estructura ColumnStatisticsTaskRunningException
- Estructura ColumnStatisticsTaskNotRunningException
- Estructura ColumnStatisticsTaskStopingException

Estructura ColumnStatisticsTaskRun

El objeto que muestra los detalles de la ejecución de las estadísticas de la columna.

Campos

- `CustomerId`: cadena UTF-8, no más de 12 bytes de largo.

  El ID de la cuenta de AWS.

- `ColumnStatisticsTaskRunId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El identificador de la ejecución de la tarea de estadísticas de columna en particular.

- `DatabaseName`: cadena UTF-8.

  La base de datos en la que se encuentra la tabla.

- `TableName`: cadena UTF-8.

  El nombre de la tabla para la que se generan las estadísticas de las columnas.

- `ColumnNameList`: matriz de cadenas UTF-8.

  Una lista de los nombres de las columnas. Si no se proporciona ninguno, todos los nombres de columna de la tabla se usarán de forma predeterminada.

- `CatalogID`: cadena de ID de catálogo, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID del Catálogo de datos donde reside la tabla. Si no se proporciona ninguno, el ID de cuenta de AWS se usará de forma predeterminada.

- `Role`: cadena UTF-8.

  El rol de IAM que el servicio asume para generar estadísticas.

- `SampleSize`: número (doble), 100 como máximo.

El porcentaje de filas que se utilizan para generar estadísticas. Si no se proporciona ninguno, la tabla completa se usará para generar estadísticas.

- `SecurityConfiguration`: cadena UTF-8 de 128 bytes de largo como máximo.

  Nombre de la configuración de seguridad que se utiliza para cifrar los registros de CloudWatch para la ejecución de la tarea de estadísticas de columnas.

- `NumberOfWorkers`: número (entero), como mínimo 1.

  El número de empleados utilizados para generar las estadísticas de las columnas. El trabajo está preconfigurado para escalar automáticamente hasta 25 instancias.

- `WorkerType`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El tipo de trabajadores que se utilizan para generar estadísticas. El valor predeterminado es `g.1x`.

- `Status`: cadena UTF-8 (valores válidos: `STARTING` | `RUNNING` | `SUCCEEDED` | `FAILED` | `STOPPED`).

  El estado de la ejecución de la tarea.

- `CreationTime`: marca temporal.

  La hora en que se creó esta tarea.

- `LastUpdated`: marca temporal.

  El último punto temporal en que se modificó esta tarea.

- `StartTime`: marca temporal.

  La hora de inicio de la tarea.

- `EndTime`: marca temporal.

  La hora de finalización de la tarea.

- `ErrorMessage`: cadena de descripción de un máximo de 2048 bytes de largo, que coincide con el URI address multi-line string pattern.

  El mensaje de error del trabajo.

- `DPUSeconds`: número (doble), cero como máximo.

  El uso de la DPU calculado en segundos para todos los trabajadores con escalado automático.

Estructura ColumnStatisticsTaskRunningException

Se produce una excepción cuando se intenta iniciar otro trabajo mientras se ejecuta un trabajo de generación de estadísticas de columnas.

Campos

- `Message`: cadena UTF-8.

  Mensaje que describe el problema.

Estructura ColumnStatisticsTaskNotRunningException

Una excepción que se produce cuando se intenta detener la ejecución de una tarea cuando no hay ninguna tarea en ejecución.

Campos

- `Message`: cadena UTF-8.

  Mensaje que describe el problema.

Estructura ColumnStatisticsTaskStopingException

Una excepción que se produce cuando se intenta detener la ejecución de una tarea.

Campos

- `Message`: cadena UTF-8.

  Mensaje que describe el problema.

Operaciones

- StartColumnStatisticsTaskRun action (Python: start_column_statistics_task_run)
- GetColumnStatisticsTaskRun action (Python: get_column_statistics_task_run)
- GetColumnStatisticsTaskRuns action (Python: get_column_statistics_task_runs)
- ListColumnStatisticsTaskRuns action (Python: list_column_statistics_task_runs)
- StopColumnStatisticsTaskRun action (Python: stop_column_statistics_task_run)

StartColumnStatisticsTaskRun action (Python: start_column_statistics_task_run)

Inicia la ejecución de una tarea de estadísticas de columnas para una tabla y columnas especificadas.

Solicitud

- `DatabaseName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la base de datos donde reside la tabla.
- `TableName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la tabla con que se van a generar estadísticas.
- `ColumnNameList`: matriz de cadenas UTF-8.

  Una lista de los nombres de las columnas para generar estadísticas. Si no se proporciona ninguno, todos los nombres de columna de la tabla se usarán de forma predeterminada.
- `Role`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El rol de IAM que el servicio asume para generar estadísticas.
- `SampleSize`: número (doble), 100 como máximo.

  El porcentaje de filas que se utilizan para generar estadísticas. Si no se proporciona ninguno, la tabla completa se usará para generar estadísticas.
- `CatalogID`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID del catálogo de datos donde reside la tabla. Si no se proporciona ninguno, el ID de cuenta de AWS se usará de forma predeterminada.
- `SecurityConfiguration`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre de la configuración de seguridad que se utiliza para cifrar los registros de CloudWatch para la ejecución de la tarea de estadísticas de columnas.

Respuesta

- `ColumnStatisticsTaskRunId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El identificador de ejecución de la tarea de estadísticas de las columnas.

Errores

- `AccessDeniedException`
- `EntityNotFoundException`
- `ColumnStatisticsTaskRunningException`
- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`
- `InvalidInputException`

GetColumnStatisticsTaskRun action (Python: get_column_statistics_task_run)

Obtenga los metadatos o la información asociados a la ejecución de una tarea, con un identificador de ejecución de la tarea determinado.

Solicitud

- `ColumnStatisticsTaskRunId`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El identificador de la ejecución de la tarea de estadísticas de columna en particular.

Respuesta

- `ColumnStatisticsTaskRun`: un objeto ColumnStatisticsTaskRun.

  Un objeto de `ColumnStatisticsTaskRun` que representa los detalles de la ejecución de las estadísticas de la columna.

Errores

- `EntityNotFoundException`
- `OperationTimeoutException`
- `InvalidInputException`

GetColumnStatisticsTaskRuns action (Python: get_column_statistics_task_runs)

Recupera información sobre todas las ejecuciones asociadas a la tabla especificada.

Solicitud

- `DatabaseName` – Obligatorio: cadena UTF-8.

  El nombre de la base de datos donde reside la tabla.
- `TableName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la tabla.
- `MaxResults`: número (entero) que no es inferior a 1 ni es superior a 1000.

  Tamaño máximo de la respuesta.
- `NextToken`: cadena UTF-8.

  Token de continuación si se trata de una llamada de continuidad.

Respuesta

- `ColumnStatisticsTaskRuns`: matriz de objetos ColumnStatisticsTaskRun.

  Una lista de las ejecuciones de tareas de estadísticas de las columnas.
- `NextToken`: cadena UTF-8.

  Un token de continuación, si todavía no se han devuelto todas las ejecuciones de tareas.

Errores

- `OperationTimeoutException`

ListColumnStatisticsTaskRuns action (Python: list_column_statistics_task_runs)

Enumera todas las ejecuciones de tareas para una cuenta en particular.

Solicitud

- `MaxResults`: número (entero) que no es inferior a 1 ni es superior a 1000.

  Tamaño máximo de la respuesta.
- `NextToken`: cadena UTF-8.

  Token de continuación si se trata de una llamada de continuidad.

Respuesta

- `ColumnStatisticsTaskRunIds`: matriz de cadenas UTF-8, con 100 cadenas como máximo.

  Una lista de los identificadores de ejecución de las tareas de estadísticas de las columnas.
- `NextToken`: cadena UTF-8.

  Un token de continuación, si todavía no se han devuelto todos los identificadores de ejecución de tareas.

Errores

- `OperationTimeoutException`

StopColumnStatisticsTaskRun action (Python: stop_column_statistics_task_run)

Detiene la ejecución de una tarea para la tabla especificada.

Solicitud

- `DatabaseName` – Obligatorio: cadena UTF-8.

  El nombre de la base de datos donde reside la tabla.
- `TableName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la tabla.

Respuesta

- Sin parámetros de respuesta.

Errores

- `EntityNotFoundException`
- `ColumnStatisticsTaskNotRunningException`
- `ColumnStatisticsTaskStoppingException`
- `OperationTimeoutException`

API del programador del rastreador

La API del programador de rastreadores describe los tipos de datos de rastreadores de AWS Glue, junto con la API para crear, eliminar, actualizar y ver listas de rastreadores.

Tipos de datos

- Estructura de programación

Estructura de programación

Objeto de programación que usa una instrucción `cron` para programar un evento.

Campos

- `ScheduleExpression`: cadena UTF-8.

  Expresión `cron` utilizada para especificar el programa (consulte Programaciones basadas en tiempo para trabajos y rastreadores). Por ejemplo, para ejecutar algo todos los días a las 12:15 UTC, especifique: `cron(15 12 * * ? *)`.
- `State`: cadena UTF-8 (valores válidos: `SCHEDULED` | `NOT_SCHEDULED` | `TRANSITIONING`).

  Estado del programa.

Operaciones

- Acción UpdateCrawlerSchedule (Python: update_crawler_schedule)

- Acción StartCrawlerSchedule (Python: start_crawler_schedule)
- Acción StopCrawlerSchedule (Python: stop_crawler_schedule)

Acción UpdateCrawlerSchedule (Python: update_crawler_schedule)

Actualiza la programación de un rastreador mediante una expresión `cron`.

Solicitud

- `CrawlerName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre del rastreador cuya programación se debe actualizar.
- `Schedule`: cadena UTF-8.

  La expresión `cron` actualizada que se utiliza para especificar el programa (consulte el tema sobre programaciones basadas en tiempo para trabajos y rastreadores). Por ejemplo, para ejecutar algo todos los días a las 12:15 UTC, especifique: `cron(15 12 * * ? *)`.

Respuesta

- Sin parámetros de respuesta.

Errores

- `EntityNotFoundException`
- `InvalidInputException`
- `VersionMismatchException`
- `SchedulerTransitioningException`
- `OperationTimeoutException`

Acción StartCrawlerSchedule (Python: start_crawler_schedule)

Cambia el estado de programación del rastreador especificado para `SCHEDULED`, salvo que el rastreador ya se esté ejecutando o el estado de programación ya esté en `SCHEDULED`.

Solicitud

- `CrawlerName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre del rastreador que se va a programar.

Respuesta

- Sin parámetros de respuesta.

Errores

- `EntityNotFoundException`
- `SchedulerRunningException`
- `SchedulerTransitioningException`
- `NoScheduleException`
- `OperationTimeoutException`

Acción StopCrawlerSchedule (Python: stop_crawler_schedule)

Establece el estado del programador del rastreador especificado en `NOT_SCHEDULED`, pero no detiene el rastreador si ya se está ejecutando.

Solicitud

- `CrawlerName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre del rastreador cuyo estado de programación se va a definir.

Respuesta

- Sin parámetros de respuesta.

Errores

- `EntityNotFoundException`
- `SchedulerNotRunningException`
- `SchedulerTransitioningException`
- `OperationTimeoutException`

API de generación automática de scripts de ETL

La API de generación de scripts de ETL describe los tipos de datos y la API de generación de scripts de ETL en AWS Glue.

Tipos de datos

- Estructura CodeGenNode
- Estructura CodeGenNodeArg
- Estructura CodeGenEdge
- Estructura de ubicación
- Estructura CatalogEntry
- Estructura MappingEntry

Estructura CodeGenNode

Representa un nodo en gráficos acíclicos dirigidos (DAG)

Campos

- `Id`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Identifier string pattern.

  Identificador de nodo que es único en el gráfico de nodos.
- `NodeType` – Obligatorio: cadena UTF-8.

  El tipo de nodo de que se trata.
- `Args` – Obligatorio: matriz de objetos CodeGenNodeArg, con 50 estructuras como máximo.

Propiedades del nodo, en forma de pares nombre-valor.

- `LineNumber`: número (entero).

  El número de la línea del nodo.

Estructura CodeGenNodeArg

Argumento o propiedad de un nodo.

Campos

- `Name` – Obligatorio: cadena UTF-8.

  El nombre del argumento o la propiedad.

- `Value` – Obligatorio: cadena UTF-8.

  El valor del argumento o la propiedad.

- `Param`: booleano.

  True si el valor se utiliza como un parámetro.

Estructura CodeGenEdge

Representa un límite direccional en un gráfico acíclico dirigido (DAG).

Campos

- `Source`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Identifier string pattern.

  El ID del nodo donde comienza el límite.

- `Target`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Identifier string pattern.

  El ID del nodo donde finaliza el límite.

- `TargetParameter`: cadena UTF-8.

  El destino del límite.

Estructura de ubicación

La ubicación de los recursos.

Campos

- `Jdbc`: matriz de objetos CodeGenNodeArg, con 50 estructuras como máximo.

  Una ubicación de JDBC.
- `S3`: matriz de objetos CodeGenNodeArg, con 50 estructuras como máximo.

  Una ubicación de Amazon Simple Storage Service (Amazon S3).
- `DynamoDB`: matriz de objetos CodeGenNodeArg, con 50 estructuras como máximo.

  Una ubicación de tabla de Amazon DynamoDB.

Estructura CatalogEntry

Especifica una definición de tabla en el AWS Glue Data Catalog.

Campos

- `DatabaseName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  La base de datos en la que se encuentran los metadatos de la tabla.
- `TableName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre de la tabla en cuestión.

Estructura MappingEntry

Define un mapeo.

Campos

- `SourceTable`: cadena UTF-8.

  El nombre de la tabla de origen.

- `SourcePath`: cadena UTF-8.

  La ruta de origen de .
- `SourceType`: cadena UTF-8.

  El tipo de origen.
- `TargetTable`: cadena UTF-8.

  La tabla de destino.
- `TargetPath`: cadena UTF-8.

  La ruta de destino.
- `TargetType`: cadena UTF-8.

  El tipo de destino.

Operaciones

- Acción CreateScript (Python: create_script)
- Acción GetDataflowGraph (Python: get_dataflow_graph)
- Acción GetMapping (Python: get_mapping)
- Acción GetPlan (Python: get_plan)

Acción CreateScript (Python: create_script)

Transforma gráficos acíclicos dirigidos (DAG) en código.

Solicitud

- `DagNodes`: matriz de objetos CodeGenNode.

  Lista de los nodos en el DAG.
- `DagEdges`: matriz de objetos CodeGenEdge.

  Lista de los límites en el DAG.
- `Language`: cadena UTF-8 (valores válidos: `PYTHON` | `SCALA`).

  El lenguaje de programación del código que genera el DAG.

Respuesta

- `PythonScript`: cadena UTF-8.

  El script de Python generado desde el DAG.
- `ScalaCode`: cadena UTF-8.

  El código Scala generado desde el DAG.

Errores

- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

Acción GetDataflowGraph (Python: get_dataflow_graph)

Transforma scripts de Python en gráficos acíclicos dirigidos (DAG).

Solicitud

- `PythonScript`: cadena UTF-8.

  El script de Python que se va a transformar.

Respuesta

- `DagNodes`: matriz de objetos CodeGenNode.

  Lista de los nodos en el DAG resultante.
- `DagEdges`: matriz de objetos CodeGenEdge.

  Lista de los límites en el DAG resultante.

Errores

- `InvalidInputException`
- `InternalServiceException`

- `OperationTimeoutException`

Acción GetMapping (Python: get_mapping)

Crea mapeos.

Solicitud

- `Source`: obligatorio: objeto CatalogEntry.

  Especifica la tabla de origen.
- `Sinks`: matriz de objetos CatalogEntry.

  Lista de las tablas de destino.
- `Location`: un objeto Ubicación.

  Parámetros para el mapeo.

Respuesta

- `Mapping` (obligatorio): una matriz de objetos MappingEntry.

  Lista de mapeos para los destinos especificados.

Errores

- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `EntityNotFoundException`

Acción GetPlan (Python: get_plan)

Obtiene código para realizar un mapeo especificado.

Solicitud

- `Mapping` (obligatorio): una matriz de objetos MappingEntry.

La lista de mapeos desde una tabla de origen a las tablas de destino.

- `Source`: obligatorio: objeto CatalogEntry.

  La tabla de origen.

- `Sinks`: matriz de objetos CatalogEntry.

  Las tablas de destino.

- `Location`: un objeto Ubicación.

  Los parámetros para el mapeo.

- `Language`: cadena UTF-8 (valores válidos: `PYTHON` | `SCALA`).

  El lenguaje de programación del código para realizar el mapeo.

- `AdditionalPlanOptionsMap`: matriz de mapas de pares clave-valor.

  Cada clave es una cadena UTF-8.

  Cada valor es una cadena UTF-8.

  Un mapa para contener parámetros opcionales de clave-valor adicionales.

  Actualmente, se soportan los siguientes pares clave-valor:

  - `inferSchema`: especifica si se debe configurar `inferSchema` como verdadero o falso para el script predeterminado generado por un trabajo de AWS Glue. Por ejemplo, para configurar `inferSchema` como verdadero, transfiera el siguiente par de clave-valor:

    ```
    --additional-plan-options-map '{"inferSchema":"true"}'
    ```

Respuesta

- `PythonScript`: cadena UTF-8.

  Script de Python para realizar el mapeo.

- `ScalaCode`: cadena UTF-8.

  Código de Scala para realizar el mapeo.

Errores

- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

API de Visual Job

La API de trabajos visuales le permite crear trabajos de integración de datos mediante la AWS Glue API desde un objeto JSON que representa la configuración visual de un AWS Glue trabajo.

`CodeGenConfigurationNodes`Se proporciona una lista de las API de creación o actualización de trabajos para registrar un DAG en AWS Glue Studio para el trabajo creado y generar el código asociado.

Tipos de datos

- CodeGenConfigurationNode estructura
- Estructura JDBC ConnectorOptions
- StreamingDataPreviewOptions estructura
- AthenaConnectorSource estructura
- Estructura JDBC ConnectorSource
- SparkConnectorSource estructura
- CatalogSource estructura
- CatalogSource Estructura de MySQL
- Estructura de PostgreSQL CatalogSource
- Estructura de OracleSQL CatalogSource
- Estructura de MicrosoftSQL ServerCatalogSource
- CatalogKinesisSource estructura
- DirectKinesisSource estructura
- KinesisStreamingSourceOptions estructura
- CatalogKafkaSource estructura

- DirectKafkaSource estructura
- KafkaStreamingSourceOptions estructura
- RedshiftSource estructura
- AmazonRedshiftSource estructura
- AmazonRedshiftNodeData estructura
- AmazonRedshiftAdvancedOption estructura
- Estructura de opción
- CatalogSource estructura S3
- SourceAdditionalOptions Estructura de S3
- CsvSource Estructura S3
- Estructura DirectJDBCSource
- DirectSourceAdditionalOptions Estructura S3
- JsonSource Estructura S3
- ParquetSource Estructura S3
- Estructura de S3 DeltaSource
- CatalogDeltaSource Estructura S3
- CatalogDeltaSource estructura
- HudiSource Estructura S3
- Estructura S3 CatalogHudiSource
- CatalogHudiSource estructura
- Estructura de DynamoDB CatalogSource
- RelationalCatalogSource estructura
- estructura JDBC ConnectorTarget
- SparkConnectorTarget estructura
- BasicCatalogTarget estructura
- CatalogTarget Estructura de MySQL
- Estructura de PostgreSQL CatalogTarget
- Estructura de OracleSQL CatalogTarget

- Estructura de Microsoft SQL ServerCatalogTarget
- RedshiftTarget estructura
- AmazonRedshiftTarget estructura
- UpsertRedshiftTargetOptions estructura
- CatalogTarget estructura S3
- GlueParquetTarget Estructura de S3
- CatalogSchemaChangePolicy estructura
- DirectTarget estructura S3
- HudiCatalogTarget Estructura S3
- Estructura S3 HudiDirectTarget
- Estructura S3 DeltaCatalogTarget
- DeltaDirectTarget Estructura S3
- DirectSchemaChangePolicy estructura
- ApplyMapping estructura
- Estructura de asignación
- SelectFields estructura
- DropFields estructura
- RenameField estructura
- Estructura Spigot
- Estructura Join
- JoinColumn estructura
- SplitFields estructura
- SelectFromCollection estructura
- FillMissingValues estructura
- Estructura Filter
- FilterExpression estructura
- FilterValue estructura
- CustomCode estructura

- Estructura SparkSQL
- SqlAlias estructura
- DropNullFields estructura
- NullCheckBoxList estructura
- NullValueField estructura
- Estructura Datatype
- Estructura Merge
- Estructura Union
- Estructura PIIDetection
- Estructura Aggregate
- DropDuplicates estructura
- GovernedCatalogTarget estructura
- GovernedCatalogSource estructura
- AggregateOperation estructura
- GlueSchema estructura
- GlueStudioSchemaColumn estructura
- GlueStudioColumn estructura
- DynamicTransform estructura
- TransformConfigParameter estructura
- EvaluateDataQuality estructura
- estructura DQ ResultsPublishingOptions
- Estructura DQ StopJobOnFailureOptions
- EvaluateDataQualityMultiFrame estructura
- Estructura de receta
- RecipeReference estructura
- SnowflakeNodeData estructura
- SnowflakeSource estructura
- SnowflakeTarget estructura
- ConnectorDataSource estructura

- ConnectorDataTarget estructura

CodeGenConfigurationNode estructura

`CodeGenConfigurationNode` enumera todos los tipos de nodos válidos. Se puede completar una y solo una de sus variables miembro.

Campos

- `AthenaConnectorSource`: un objeto AthenaConnectorSource.

  Especifica un conector a un origen de datos de Amazon Athena.
- `JDBCConnectorSource`: un objeto JDBC ConnectorSource.

  Especifica un conector a un origen de datos JDBC.
- `SparkConnectorSource`: un objeto SparkConnectorSource.

  Especifica un conector a un origen de datos de Apache Spark.
- `CatalogSource`: un objeto CatalogSource.

  Especifica un banco de datos en el catálogo AWS Glue de datos.
- `RedshiftSource`: un objeto RedshiftSource.

  Especifica un almacén de datos de Amazon Redshift.
- `S3CatalogSource`: un objeto S3 CatalogSource.

  Especifica un almacén de datos de Amazon S3 en el catálogo AWS Glue de datos.
- `S3CsvSource`: un objeto S3 CsvSource.

  Especifica un almacén de datos de valores separados por comas (CSV) almacenado en Amazon S3.
- `S3JsonSource`: un objeto S3 JsonSource.

  Especifica un almacén de datos JSON almacenado en Amazon S3.
- `S3ParquetSource`: un objeto S3 ParquetSource.

  Especifica un almacén de datos de Apache Parquet almacenado en Amazon S3.
- `RelationalCatalogSource`: un objeto RelationalCatalogSource.

Especifica un almacén de datos del catálogo relacional en el catálogo AWS Glue de datos.

- `DynamoDBCatalogSource`: un objeto DynamoDB CatalogSource.

  Especifica un banco de datos del catálogo de DynamoDBC en el AWS Glue catálogo de datos.

- `JDBCConnectorTarget`: un objeto JDBC ConnectorTarget.

  Especifica un destino de datos que escribe en Amazon S3 en el almacenamiento en columnas de Apache Parquet.

- `SparkConnectorTarget`: un objeto SparkConnectorTarget.

  Especifica un destino que utiliza un conector de Apache Spark.

- `CatalogTarget`: un objeto BasicCatalogTarget.

  Especifica un destino que utiliza una tabla del catálogo AWS Glue de datos.

- `RedshiftTarget`: un objeto RedshiftTarget.

  Especifica un destino que utiliza Amazon Redshift.

- `S3CatalogTarget`: un objeto S3 CatalogTarget.

  Especifica un destino de datos que escribe en Amazon S3 mediante el catálogo AWS Glue de datos.

- `S3GlueParquetTarget`: un objeto S3 GlueParquetTarget.

  Especifica un destino de datos que escribe en Amazon S3 en el almacenamiento en columnas de Apache Parquet.

- `S3DirectTarget`: un objeto S3 DirectTarget.

  Especifica un destino de datos que escribe en Amazon S3.

- `ApplyMapping`: un objeto ApplyMapping.

  Especifica una transformación que asigna claves de propiedad de datos en el origen de datos a claves de propiedad de datos en el destino de datos. Puede cambiar el nombre de las claves, modificar los tipos de datos de las claves y elegir las claves que desea descartar del conjunto de datos.

- `SelectFields`: un objeto SelectFields.

  Especifica una transformación que elige las claves de propiedad de datos que desea conservar.

- `DropFields`: un objeto DropFields.

  Especifica una transformación que elige las claves de propiedad de datos que desea eliminar.
- `RenameField`: un objeto RenameField.

  Especifica una transformación que cambia el nombre de una única clave de propiedad de datos.
- `Spigot`: un objeto Spigot.

  Especifica una transformación que escribe ejemplos de los datos en un bucket de Amazon S3.
- `Join`: un objeto Join.

  Especifica una transformación que une dos conjuntos de datos en uno mediante una frase de comparación en las claves de propiedad de datos especificadas. Puede utilizar combinaciones interna, externa, izquierda, derecha, semicombinación izquierda y anticombinación izquierda.
- `SplitFields`: un objeto SplitFields.

  Especifica una transformación que divide las claves de propiedad de datos en dos `DynamicFrames`. La salida es una recopilación de `DynamicFrames`: uno con las claves de propiedad de datos seleccionadas y el otro con las claves de propiedad de datos restantes.
- `SelectFromCollection`: un objeto SelectFromCollection.

  Especifica una transformación que elige un `DynamicFrame` de una recopilación de `DynamicFrames`. El resultado es el `DynamicFrame` seleccionado
- `FillMissingValues`: un objeto FillMissingValues.

  Especifica una transformación que localiza registros en el conjunto de datos que tienen valores faltantes y agrega un nuevo campo con un valor determinado por imputación. El conjunto de datos de entrada se utiliza para formar al modelo de machine learning que determina cuál debe ser el valor que falta.
- `Filter`: un objeto Filtro.

  Especifica una transformación que divide un conjunto de datos en dos, en función de una condición de filtro.
- `CustomCode`: un objeto CustomCode.

  Especifica una transformación que utiliza el código personalizado que proporciona el usuario para llevar a cabo la transformación de datos. El resultado es una colección de DynamicFrames.
- `SparkSQL`: un objeto SparkSQL.

Especifica una transformación en la que se ingresa una consulta SQL mediante la sintaxis de Spark SQL para transformar los datos. La salida es un único `DynamicFrame`.

- `DirectKinesisSource`: un objeto DirectKinesisSource.

  Especifica un origen de datos directo de Amazon Kinesis.

- `DirectKafkaSource`: un objeto DirectKafkaSource.

  Especifica un almacén de datos de Apache Kafka.

- `CatalogKinesisSource`: un objeto CatalogKinesisSource.

  Especifica una fuente de datos de Kinesis en el catálogo de AWS Glue datos.

- `CatalogKafkaSource`: un objeto CatalogKafkaSource.

  Especifica un almacén de datos de Apache Kafka en Data Catalog.

- `DropNullFields`: un objeto DropNullFields.

  Especifica una transformación que elimina columnas del conjunto de datos si todos los valores de la columna son “nulos”. De forma predeterminada, AWS Glue Studio reconocerá los objetos nulos, pero algunos valores, como las cadenas vacías, las cadenas «nulas», los enteros -1 u otros marcadores de posición, como ceros, no se reconocen automáticamente como nulos.

- `Merge`: un objeto Merge.

  Especifica una transformación que fusiona un `DynamicFrame` con una instancia provisional de `DynamicFrame` en función de las claves principales especificadas para identificar registros. Los registros duplicados (registros con las mismas claves principales) no se eliminan.

- `Union`: un objeto Unión.

  Especifica una transformación que combina las filas de dos o más conjuntos de datos en un único resultado.

- `PIIDetection`: un objeto PIIDetection.

  Especifica una transformación que identifica, elimina o enmascara datos PII.

- `Aggregate`: un objeto Agregado.

  Especifica una transformación que agrupa las filas según los campos elegidos y calcula el valor agregado mediante una función especificada.

- `DropDuplicates`: un objeto DropDuplicates.

Especifica una transformación que elimina las filas de datos repetidos de un conjunto de datos.

- `GovernedCatalogTarget`: un objeto GovernedCatalogTarget.

  Especifica un destino de datos que escribe en un catálogo gobernado.

- `GovernedCatalogSource`: un objeto GovernedCatalogSource.

  Especifica un origen de datos en un Data Catalog gobernado.

- `MicrosoftSQLServerCatalogSource`: un objeto Microsoft SQL ServerCatalogSource.

  Especifica un origen de datos de Microsoft SQL server en AWS Glue Data Catalog.

- `MySQLCatalogSource`: un objeto MySQL CatalogSource.

  Especifica una fuente de datos MySQL en el catálogo AWS Glue de datos.

- `OracleSQLCatalogSource`: un objeto OracleSQL CatalogSource.

  Especifica una fuente de datos de Oracle en el catálogo AWS Glue de datos.

- `PostgreSQLCatalogSource`: un objeto PostgresQL CatalogSource.

  Especifica una fuente de datos de PostgresSQL en AWS Glue el catálogo de datos.

- `MicrosoftSQLServerCatalogTarget`: un objeto Microsoft SQL ServerCatalogTarget.

  Especifica un destino que utiliza Microsoft SQL.

- `MySQLCatalogTarget`: un objeto MySQL CatalogTarget.

  Especifica un destino que utiliza MySQL.

- `OracleSQLCatalogTarget`: un objeto OracleSQL CatalogTarget.

  Especifica un destino que utiliza Oracle SQL.

- `PostgreSQLCatalogTarget`: un objeto PostgresQL CatalogTarget.

  Especifica un destino que utiliza PostgreSQL.

- `DynamicTransform`: un objeto DynamicTransform.

  Especifica una transformación visual personalizada que haya creado un usuario.

- `EvaluateDataQuality`: un objeto EvaluateDataQuality.

  Especifica los criterios de evaluación de la calidad de los datos.

- `S3CatalogHudiSource`: un objeto S3 CatalogHudiSource.

  Especifica una fuente de datos de Hudi que está registrada en el AWS Glue catálogo de datos. La fuente de datos debe almacenarse en Amazon S3.

- `CatalogHudiSource`: un objeto CatalogHudiSource.

  Especifica una fuente de datos de Hudi que está registrada en el catálogo de AWS Glue datos.

- `S3HudiSource`: un objeto S3 HudiSource.

  Especifica una fuente de datos Hudi almacenada en. Amazon S3

- `S3HudiCatalogTarget`: un objeto S3 HudiCatalogTarget.

  Especifica un destino que escribe en una fuente de datos de Hudi del catálogo de AWS Glue datos.

- `S3HudiDirectTarget`: un objeto S3 HudiDirectTarget.

  Especifica un destino que escribe en una fuente de datos de Hudi. Amazon S3

- `S3CatalogDeltaSource`: un objeto S3 CatalogDeltaSource.

  Especifica una fuente de datos de Delta Lake que está registrada en el catálogo de AWS Glue datos. La fuente de datos debe almacenarse en Amazon S3.

- `CatalogDeltaSource`: un objeto CatalogDeltaSource.

  Especifica una fuente de datos de Delta Lake que está registrada en el catálogo AWS Glue de datos.

- `S3DeltaSource`: un objeto S3 DeltaSource.

  Especifica una fuente de datos de Delta Lake almacenada en Amazon S3.

- `S3DeltaCatalogTarget`: un objeto S3 DeltaCatalogTarget.

  Especifica un destino que escribe en una fuente de datos de Delta Lake del catálogo AWS Glue de datos.

- `S3DeltaDirectTarget`: un objeto S3 DeltaDirectTarget.

  Especifica un destino que escribe en una fuente de datos de Delta Lake Amazon S3.

- `AmazonRedshiftSource`: un objeto AmazonRedshiftSource.

  Especifica un destino que escribe en un origen de datos en Amazon Redshift.

- `AmazonRedshiftTarget`: un objeto AmazonRedshiftTarget.

  Especifica un destino que escribe en un destino de datos en Amazon Redshift.

- `EvaluateDataQualityMultiFrame`: un objeto EvaluateDataQualityMultiFrame.

  Especifica los criterios de evaluación de la calidad de los datos. Permite múltiples datos de entrada y devuelve una colección de marcos dinámicos.

- `Recipe`: un objeto Receta.

  Especifica un nodo de AWS Glue DataBrew receta.

- `SnowflakeSource`: un objeto SnowflakeSource.

  Especifica un origen de datos de Snowflake.

- `SnowflakeTarget`: un objeto SnowflakeTarget.

  Especifica un destino que escribe en un origen de datos de Snowflake.

- `ConnectorDataSource`: un objeto ConnectorDataSource.

  Especifica una fuente generada con opciones de conexión estándar.

- `ConnectorDataTarget`: un objeto ConnectorDataTarget.

  Especifica un destino generado con opciones de conexión estándar.

Estructura JDBC ConnectorOptions

Opciones de conexión adicionales para el conector.

Campos

- `FilterPredicate`: cadena UTF-8 que coincide con el Custom string pattern #40.

  Cláusula de condición adicional para filtrar datos desde el origen. Por ejemplo:

  ```
  BillingCity='Mountain View'
  ```

  Cuando se utiliza una consulta en lugar de una tabla, se debe validar que la consulta funciona con el `filterPredicate` especificado.

- `PartitionColumn`: cadena UTF-8 que coincide con el Custom string pattern #40.

El nombre de una columna entera que se utiliza para particionar. Esta opción solo funciona cuando está incluida con `lowerBound`, `upperBound` y `numPartitions`. Esta opción funciona de la misma manera que en el lector JDBC de Spark SQL.

- `LowerBound`: número (largo), cero como máximo.

  El valor mínimo de `partitionColumn` que se utiliza para decidir el intervalo de partición.

- `UpperBound`: número (largo), cero como máximo.

  El valor máximo de `partitionColumn` que se utiliza para decidir el intervalo de partición.

- `NumPartitions`: número (largo), cero como máximo.

  El número de particiones. Este valor, junto con `lowerBound` (inclusive) y `upperBound` (exclusivo), forma intervalos de partición para expresiones de la cláusula `WHERE` generadas, que se utilizan para dividir la `partitionColumn`.

- `JobBookmarkKeys`: matriz de cadenas UTF-8.

  El nombre de las claves favoritas de trabajo en las que se ordenará.

- `JobBookmarkKeysSortOrder`: cadena UTF-8 que coincide con el Custom string pattern #40.

  Especifica el orden de clasificación ascendente o descendente.

- `DataTypeMapping`: matriz de mapas de pares clave-valor.

  Cada clave es una cadena UTF-8 (valores válidos: `ARRAY` | `BIGINT` | `BINARY` | `BIT` | `BLOB` | `BOOLEAN` | `CHAR` | `CLOB` | `DATALINK` | `DATE` | `DECIMAL` | `DISTINCT` | `DOUBLE` | `FLOAT` | `INTEGER` | `JAVA_OBJECT` | `LONGNVARCHAR` | `LONGVARBINARY` | `LONGVARCHAR` | `NCHAR` | `NCLOB` | `NULL` | `NUMERIC` | `NVARCHAR` | `OTHER` | `REAL` | `REF` | `REF_CURSOR` | `ROWID` | `SMALLINT` | `SQLXML` | `STRUCT` | `TIME` | `TIME_WITH_TIMEZONE` | `TIMESTAMP` | `TIMESTAMP_WITH_TIMEZONE` | `TINYINT` | `VARBINARY` | `VARCHAR`).

  Cada valor es una cadena UTF-8 (valores válidos: `DATE` | `STRING` | `TIMESTAMP` | `INT` | `FLOAT` | `LONG` | `BIGDECIMAL` | `BYTE` | `SHORT` | `DOUBLE`).

  Asignación de tipos de datos personalizada, que crea una asignación a partir de un tipo de datos JDBC a un tipo de datos de AWS Glue . Por ejemplo, la opción `"dataTypeMapping":{"FLOAT":"STRING"}` asigna campos de datos de tipo JDBC `FLOAT` al `String` tipo Java llamando al `ResultSet.getString()` método del controlador y lo usa para crear el registro. AWS Glue Cada controlador implementa el objeto `ResultSet`, por lo que el comportamiento es

específico del controlador que se utiliza. Consulte la documentación del controlador JDBC para comprender cómo el controlador realiza las conversiones.

StreamingDataPreviewOptions estructura

Especifica las opciones relacionadas con la versión preliminar de datos para ver una muestra de los datos.

Campos

- `PollingTime`: número (largo), como mínimo 10.

  El tiempo de sondeo en milisegundos.
- `RecordPollingLimit`: número (largo), como mínimo 1.

  El límite del número de registros sondeados.

AthenaConnectorSource estructura

Especifica un conector a un origen de datos de Amazon Athena.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del origen de datos.
- `ConnectionName`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de la conexión asociada al conector.
- `ConnectorName`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de un conector que ayuda a acceder al almacén de datos de AWS Glue Studio.
- `ConnectionType`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El tipo de conexión, como marketplace.athena o custom.athena, que designa una conexión a un almacén de datos de Amazon Athena.
- `ConnectionTable`: cadena UTF-8 que coincide con el Custom string pattern #41.

  El nombre de la tabla en el origen de datos.

- `SchemaName`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre del grupo de registro de CloudWatch de lectura. Por ejemplo, `/aws-glue/jobs/output`.
- `OutputSchemas`: matriz de objetos GlueSchema.

  Especifica el esquema de datos del origen de Athena personalizado.

Estructura JDBC ConnectorSource

Especifica un conector a un origen de datos JDBC.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del origen de datos.
- `ConnectionName`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de la conexión asociada al conector.
- `ConnectorName`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de un conector que ayuda a acceder al almacén de datos de Studio. AWS Glue
- `ConnectionType`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El tipo de conexión, como marketplace.jdbc o custom.jdbc, que designa una conexión a un almacén de datos JDBC.
- `AdditionalOptions`: un objeto JDBC ConnectorOptions.

  Opciones de conexión adicionales para el conector.
- `ConnectionTable`: cadena UTF-8 que coincide con el Custom string pattern #41.

  El nombre de la tabla en el origen de datos.
- `Query`: cadena UTF-8 que coincide con el Custom string pattern #42.

  La tabla o consulta SQL de la que se obtienen los datos. Puede especificar `ConnectionTable` o `query`, pero no ambos.
- `OutputSchemas`: matriz de objetos GlueSchema.

Especifica el esquema de datos del origen de JDBC personalizado.

SparkConnectorSource estructura

Especifica un conector a un origen de datos de Apache Spark.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del origen de datos.
- `ConnectionName`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de la conexión asociada al conector.
- `ConnectorName`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de un conector que ayuda a acceder al almacén de datos de AWS Glue Studio.
- `ConnectionType`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El tipo de conexión, como marketplace.spark o custom.spark, que designa una conexión a un almacén de datos de Apache Spark.
- `AdditionalOptions`: matriz de mapas de pares clave-valor.

  Cada clave es una cadena UTF-8 que coincide con el Custom string pattern #40.

  Cada valor es una cadena UTF-8 que coincide con el Custom string pattern #40.

  Opciones de conexión adicionales para el conector.
- `OutputSchemas`: matriz de objetos GlueSchema.

  Especifica el esquema de datos del origen de spark personalizado.

CatalogSource estructura

Especifica un banco de datos en el catálogo AWS Glue de datos.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

El nombre del almacén de datos.

- `Database`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de la base de datos de lectura.

- `Table`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de la tabla en la base de datos de lectura.

CatalogSource Estructura de MySQL

Especifica una fuente de datos MySQL en el catálogo AWS Glue de datos.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del origen de datos.

- `Database`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de la base de datos de lectura.

- `Table`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de la tabla en la base de datos de lectura.

Estructura de PostgreSQL CatalogSource

Especifica una fuente de datos de PostgresSQL en AWS Glue el catálogo de datos.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del origen de datos.

- `Database`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de la base de datos de lectura.

- `Table`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

El nombre de la tabla en la base de datos de lectura.

Estructura de OracleSQL CatalogSource

Especifica una fuente de datos de Oracle en el AWS Glue catálogo de datos.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del origen de datos.
- `Database`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de la base de datos de lectura.
- `Table`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de la tabla en la base de datos de lectura.

Estructura de MicrosoftSQL ServerCatalogSource

Especifica un origen de datos de Microsoft SQL server en AWS Glue Data Catalog.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del origen de datos.
- `Database`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de la base de datos de lectura.
- `Table`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de la tabla en la base de datos de lectura.

CatalogKinesisSource estructura

Especifica una fuente de datos de Kinesis en el catálogo de AWS Glue datos.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del origen de datos.
- `WindowSize`: número (entero), cero como máximo.

  La cantidad de tiempo que se debe dedicar al procesamiento de cada microlote.
- `DetectSchema`: booleano.

  Si se debe determinar automáticamente el esquema a partir de los datos entrantes.
- `Table`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de la tabla en la base de datos de lectura.
- `Database`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de la base de datos de lectura.
- `StreamingOptions`: un objeto KinesisStreamingSourceOptions.

  Opciones adicionales para el origen de datos de streaming de Kinesis.
- `DataPreviewOptions`: un objeto StreamingDataPreviewOptions.

  Opciones adicionales para la versión preliminar de datos.

DirectKinesisSource estructura

Especifica un origen de datos directo de Amazon Kinesis.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del origen de datos.
- `WindowSize`: número (entero), cero como máximo.

  La cantidad de tiempo que se debe dedicar al procesamiento de cada microlote.
- `DetectSchema`: booleano.

  Si se debe determinar automáticamente el esquema a partir de los datos entrantes.

- `StreamingOptions`: un objeto KinesisStreamingSourceOptions.

  Opciones adicionales para el origen de datos de streaming de Kinesis.

- `DataPreviewOptions`: un objeto StreamingDataPreviewOptions.

  Opciones adicionales para la versión preliminar de datos.

KinesisStreamingSourceOptions estructura

Opciones adicionales para el origen de datos de streaming de Amazon Kinesis.

Campos

- `EndpointUrl`: cadena UTF-8 que coincide con el Custom string pattern #40.

  La URL del punto de conexión de Kinesis.

- `StreamName`: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre del flujo de datos de Kinesis.

- `Classification`: cadena UTF-8 que coincide con el Custom string pattern #40.

  Una clasificación opcional.

- `Delimiter`: cadena UTF-8 que coincide con el Custom string pattern #40.

  Especifica el carácter delimitador.

- `StartingPosition`: cadena UTF-8 (valores válidos: `latest="LATEST"` | `trim_horizon="TRIM_HORIZON"` | `earliest="EARLIEST"` | `timestamp="TIMESTAMP"`).

  La posición inicial en el flujo de datos de Kinesis para leer los datos. Los valores posibles son `"latest"`, `"trim_horizon"`, `"earliest"` o una cadena de marca de tiempo en formato UTC en el patrón `yyyy-mm-ddTHH:MM:SSZ` (donde `Z` representa un desplazamiento de zona horaria UTC con un +/-. Por ejemplo, “04-04-2023 T 08:00:00-04:00”). El valor predeterminado es `"latest"`.

  Nota: El uso de un valor que sea una cadena de marca de tiempo en formato UTC para «StartingPosition» solo se admite en la AWS Glue versión 4.0 o posterior.

- `MaxFetchTimeInMs`: número (largo), cero como máximo.

El tiempo máximo que le tomó al ejecutor del trabajo leer los registros del lote actual en el flujo de datos de Kinesis, especificado en milisegundos (ms). Pueden realizarse varias llamadas a la API de `GetRecords` durante este tiempo. El valor predeterminado es `1000`.

- `MaxFetchRecordsPerShard`: número (largo), cero como máximo.

  El número máximo de registros que se recuperará por partición en el flujo de datos de Kinesis por microlote. Nota: El cliente puede exceder este límite si el trabajo de streaming ya leyó registros adicionales de Kinesis (en la misma llamada de obtención de registros). Si `MaxFetchRecordsPerShard` tiene que ser preciso, entonces tiene que ser un múltiplo de `MaxRecordPerRead`. El valor predeterminado es `100000`.

- `MaxRecordPerRead`: número (largo), cero como máximo.

  El número máximo de registros que se recuperará del flujo de datos de Kinesis en cada operación getRecords. El valor predeterminado es `10000`.

- `AddIdleTimeBetweenReads`: booleano.

  Agrega un retardo de tiempo entre dos operaciones getRecords consecutivas. El valor predeterminado es `"False"`. Esta opción sólo se puede configurar para Glue versión 2.0 y superior.

- `IdleTimeBetweenReadsInMs`: número (largo), cero como máximo.

  El retardo de tiempo mínimo entre dos operaciones getRecords consecutivas, especificado en ms. El valor predeterminado es `1000`. Esta opción sólo se puede configurar para Glue versión 2.0 y superior.

- `DescribeShardInterval`: número (largo), cero como máximo.

  El intervalo de tiempo mínimo entre dos llamadas a la ListShards API para que el script considere la posibilidad de repartirlo. El valor predeterminado es `1s`.

- `NumRetries`: número (entero), cero como máximo.

  El número máximo de reintentos para las solicitudes de la API de Kinesis Data Streams. El valor predeterminado es `3`.

- `RetryIntervalMs`: número (largo), cero como máximo.

  El periodo de enfriamiento (especificado en ms) antes de volver a intentar la llamada a la API de Kinesis Data Streams. El valor predeterminado es `1000`.

- `MaxRetryIntervalMs`: número (largo), cero como máximo.

  El periodo de enfriamiento máximo (especificado en ms) entre dos intentos de llamada a la API de Kinesis Data Streams. El valor predeterminado es `10000`.

- `AvoidEmptyBatches`: booleano.

  Evita crear un trabajo de microlotes vacío al comprobar si hay datos no leídos en el flujo de datos de Kinesis antes de que se inicie el lote. El valor predeterminado es `"False"`.

- `StreamArn`: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de recurso de Amazon (ARN) del flujo de datos de Kinesis.

- `RoleArn`: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de recurso de Amazon (ARN) del rol que se va a asumir mediante AWS Security Token Service (AWS STS). Este rol debe tener permisos para describir o leer operaciones de registros del flujo de datos de Kinesis. Debe utilizar este parámetro para acceder a un flujo de datos de otra cuenta. Se utiliza junto con `"awsSTSSessionName"`.

- `RoleSessionName`: cadena UTF-8 que coincide con el Custom string pattern #40.

  Un identificador para la sesión que asume el rol mediante AWS STS. Debe utilizar este parámetro para acceder a un flujo de datos de otra cuenta. Se utiliza junto con `"awsSTSRoleARN"`.

- `AddRecordTimestamp`: cadena UTF-8 que coincide con el Custom string pattern #40.

  Cuando esta opción se establece en “true”, la salida de datos contendrá una columna adicional denominada “__src_timestamp” que indica la hora en la que el flujo recibió el registro correspondiente. El valor predeterminado es "false". Esta opción se admite en la AWS Glue versión 4.0 o posterior.

- `EmitConsumerLagMetrics`: cadena UTF-8 que coincide con el Custom string pattern #40.

  Si esta opción se establece en «true», para cada lote, emitirá las métricas correspondientes al período comprendido entre el registro más antiguo recibido por la transmisión y el momento en AWS Glue que llegue CloudWatch. El nombre de la métrica es «glue.driver.streaming». maxConsumerLagInMs». El valor predeterminado es "false". Esta opción es compatible con la versión 4.0 o posterior de AWS Glue .

- `StartingTimestamp`: cadena UTF-8.

  La marca de tiempo del registro del flujo de datos de Kinesis desde la que empezar a leer los datos. Los valores posibles son una cadena de marca de tiempo en formato UTC en el patrón

`yyyy-mm-ddTHH:MM:SSZ` (donde Z representa un desplazamiento de zona horaria UTC con un +/-. Por ejemplo, “2023-04-04T08:00:00+08:00”).

CatalogKafkaSource estructura

Especifica un almacén de datos de Apache Kafka en Data Catalog.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del almacén de datos.
- `WindowSize`: número (entero), cero como máximo.

  La cantidad de tiempo que se debe dedicar al procesamiento de cada microlote.
- `DetectSchema`: booleano.

  Si se debe determinar automáticamente el esquema a partir de los datos entrantes.
- `Table`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de la tabla en la base de datos de lectura.
- `Database`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de la base de datos de lectura.
- `StreamingOptions`: un objeto KafkaStreamingSourceOptions.

  Especifica las opciones de streaming.
- `DataPreviewOptions`: un objeto StreamingDataPreviewOptions.

  Especifica las opciones relacionadas con la versión preliminar de datos para ver una muestra de los datos.

DirectKafkaSource estructura

Especifica un almacén de datos de Apache Kafka.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

El nombre del almacén de datos.

- `StreamingOptions`: un objeto KafkaStreamingSourceOptions.

  Especifica las opciones de streaming.

- `WindowSize`: número (entero), cero como máximo.

  La cantidad de tiempo que se debe dedicar al procesamiento de cada microlote.

- `DetectSchema`: booleano.

  Si se debe determinar automáticamente el esquema a partir de los datos entrantes.

- `DataPreviewOptions`: un objeto StreamingDataPreviewOptions.

  Especifica las opciones relacionadas con la versión preliminar de datos para ver una muestra de los datos.

KafkaStreamingSourceOptions estructura

Opciones adicionales para streaming.

Campos

- `BootstrapServers`: cadena UTF-8 que coincide con el Custom string pattern #40.

  Una lista de direcciones URL de servidor Bootstrap, por ejemplo, como `b-1.vpc-test-2.o4q88o.c6.kafka.us-east-1.amazonaws.com:9094`. Esta opción debe especificarse en la llamada a la API o definirse en los metadatos de la tabla en el Data Catalog.

- `SecurityProtocol`: cadena UTF-8 que coincide con el Custom string pattern #40.

  El protocolo utilizado para la comunicación con los agentes. Los valores posibles son `"SSL"` o.`"PLAINTEXT"`

- `ConnectionName`: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de la conexión.

- `TopicName`: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre del tema como se especifica en Apache Kafka. Debe especificar al menos una opción entre `"topicName"`, `"assign"` o `"subscribePattern"`.

- `Assign`: cadena UTF-8 que coincide con el Custom string pattern #40.

  Las `TopicPartitions` específicas que se utilizarán. Debe especificar al menos una opción entre `"topicName"`, `"assign"` o `"subscribePattern"`.

- `SubscribePattern`: cadena UTF-8 que coincide con el Custom string pattern #40.

  Una cadena de expresiones regulares de Java que identifica la lista de temas a la que desea suscribirse. Debe especificar al menos una opción entre `"topicName"`, `"assign"` o `"subscribePattern"`.

- `Classification`: cadena UTF-8 que coincide con el Custom string pattern #40.

  Una clasificación opcional.

- `Delimiter`: cadena UTF-8 que coincide con el Custom string pattern #40.

  Especifica el carácter delimitador.

- `StartingOffsets`: cadena UTF-8 que coincide con el Custom string pattern #40.

  La posición inicial en el tema de Kafka para leer los datos. Los valores posibles son `"earliest"` o.`"latest"` El valor predeterminado es `"latest"`.

- `EndingOffsets`: cadena UTF-8 que coincide con el Custom string pattern #40.

  El punto de conexión cuando finaliza una consulta por lotes. Los valores posibles son `"latest"` o una cadena JSON que especifica una compensación final para cada `TopicPartition`.

- `PollTimeoutMs`: número (largo), cero como máximo.

  El tiempo de espera en milisegundos para sondear datos de Kafka en ejecutores de trabajos de Spark. El valor predeterminado es `512`.

- `NumRetries`: número (entero), cero como máximo.

  El número de veces que se reintenta antes de no obtener las compensaciones de Kafka. El valor predeterminado es `3`.

- `RetryIntervalMs`: número (largo), cero como máximo.

  El tiempo en milisegundos para esperar antes de volver a intentar obtener compensaciones Kafka. El valor predeterminado es `10`.

- `MaxOffsetsPerTrigger`: número (largo), cero como máximo.

El límite de velocidad en el número máximo de compensaciones que se procesan por intervalo de desencadenador. El número total de compensaciones especificado se divide de forma proporcional entre `topicPartitions` de diferentes volúmenes. El valor predeterminado es nulo, lo que significa que el consumidor lee todos las compensaciones hasta la última compensación conocida.

- `MinPartitions`: número (entero), cero como máximo.

  El número mínimo deseado de particiones para leer desde Kafka. El valor predeterminado es nulo, lo que significa que el número de particiones de Spark es igual al número de particiones de Kafka.

- `IncludeHeaders`: booleano.

  Si se incluyen los encabezados de Kafka. Cuando la opción se establece en “true”, la salida de datos contendrá una columna adicional denominada “glue_streaming_kafka_headers” con el tipo `Array[Struct(key: String, value: String)]`. El valor predeterminado es “falso”. Esta opción solo está disponible en AWS Glue la versión 3.0 o posterior.

- `AddRecordTimestamp`: cadena UTF-8 que coincide con el Custom string pattern #40.

  Cuando esta opción se establece en “true”, la salida de datos contendrá una columna adicional denominada “__src_timestamp” que indica la hora en la que el tema recibió el registro correspondiente. El valor predeterminado es "false". Esta opción es compatible con la AWS Glue versión 4.0 o posterior.

- `EmitConsumerLagMetrics`: cadena UTF-8 que coincide con el Custom string pattern #40.

  Si esta opción se establece en «true», para cada lote, emitirá las métricas correspondientes al período comprendido entre el registro más antiguo recibido por el tema y el momento en AWS Glue que llegue CloudWatch. El nombre de la métrica es «glue.driver.streaming». maxConsumerLagInMs». El valor predeterminado es "false". Esta opción es compatible con la versión 4.0 o posterior de AWS Glue .

- `StartingTimestamp`: cadena UTF-8.

  La marca de tiempo del registro en el tema de Kafka desde el que empezar a leer los datos. Los valores posibles son una cadena de marca de tiempo en formato UTC en el patrón `yyyy-mm-ddTHH:MM:SSZ` (donde Z representa un desplazamiento de zona horaria UTC con un +/-. Por ejemplo, “2023-04-04T08:00:00+08:00”).

  Solo se debe configurar una de `StartingTimestamp` o `StartingOffsets`.

RedshiftSource estructura

Especifica un almacén de datos de Amazon Redshift.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del almacén de datos de Amazon Redshift.
- `Database`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  La base de datos de lectura.
- `Table`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  La tabla de base de datos de lectura.
- `RedshiftTmpDir`: cadena UTF-8 que coincide con el Custom string pattern #40.

  La ruta de Amazon S3 donde se pueden almacenar datos temporales al copiar desde la base de datos.
- `TmpDirIAMRole`: cadena UTF-8 que coincide con el Custom string pattern #40.

  El rol de IAM con permisos.

AmazonRedshiftSource estructura

Especifica el origen de Amazon Redshift.

Campos

- `Name`: cadena UTF-8 que coincide con el Custom string pattern #43.

  Nombre del origen de Amazon Redshift.
- `Data`: un objeto AmazonRedshiftNodeData.

  Especifica los datos del nodo de origen de Amazon Redshift.

AmazonRedshiftNodeData estructura

Especifica un nodo de Amazon Redshift.

Campos

- `AccessType`: cadena UTF-8 que coincide con el Custom string pattern #39.

  El tipo de acceso para la conexión de Redshift. Puede ser una conexión directa o una conexión de catálogo.

- `SourceType`: cadena UTF-8 que coincide con el Custom string pattern #39.

  El tipo de origen para especificar si una tabla específica es el origen o una consulta personalizada.

- `Connection`: un objeto Opción.

  La AWS Glue conexión al clúster de Redshift.

- `Schema`: un objeto Opción.

  El nombre del esquema de Redshift cuando se trabaja con una conexión directa.

- `Table`: un objeto Opción.

  El nombre de la tabla de Redshift cuando se trabaja con una conexión directa.

- `CatalogDatabase`: un objeto Opción.

  El nombre de la base de AWS Glue datos del catálogo de datos cuando se trabaja con un catálogo de datos.

- `CatalogTable`: un objeto Opción.

  El nombre de la tabla del catálogo de AWS Glue datos cuando se trabaja con un catálogo de datos.

- `CatalogRedshiftSchema`: cadena UTF-8.

  El nombre del esquema de Redshift cuando se trabaja con un catálogo de datos.

- `CatalogRedshiftTable`: cadena UTF-8.

  La tabla de base de datos de lectura.

- `TempDir`: cadena UTF-8 que coincide con el Custom string pattern #40.

  La ruta de Amazon S3 donde se pueden almacenar datos temporales al copiar desde la base de datos.

- `IamRole`: un objeto Opción.

Opcional. El nombre del rol que se utiliza al conectarse a S3. El rol de IAM será la función del trabajo de forma predeterminada si se deja en blanco.

- `AdvancedOptions`: matriz de objetos AmazonRedshiftAdvancedOption.

  Valores opcionales al conectarse al clúster de Redshift.

- `SampleQuery`: cadena UTF-8.

  El SQL que se utiliza para obtener los datos de una fuente de Redshift cuando se SourceType trata de una «consulta».

- `PreAction`: cadena UTF-8.

  El SQL utilizado antes de ejecutar un comando MERGE o APPEND con upsert.

- `PostAction`: cadena UTF-8.

  El SQL utilizado antes de ejecutar un comando MERGE o APPEND con upsert.

- `Action`: cadena UTF-8.

  Especifica cómo se escribirá en un clúster de Redshift.

- `TablePrefix`: cadena UTF-8 que coincide con el Custom string pattern #39.

  Especifica el prefijo a una tabla.

- `Upsert`: booleano.

  La acción utilizada en Redshift se hunde al realizar un APPEND.

- `MergeAction`: cadena UTF-8 que coincide con el Custom string pattern #39.

  La acción que se utiliza para determinar cómo se gestionará un MERGE en un receptor de Redshift.

- `MergeWhenMatched`: cadena UTF-8 que coincide con el Custom string pattern #39.

  La acción que se utiliza para determinar cómo se gestionará un MERGE en un receptor de Redshift cuando un registro existente coincida con un registro nuevo.

- `MergeWhenNotMatched`: cadena UTF-8 que coincide con el Custom string pattern #39.

  La acción que se utiliza para determinar cómo se gestionará un MERGE en un receptor de Redshift cuando un registro existente no coincida con un registro nuevo.

- `MergeClause`: cadena UTF-8.

El SQL utilizado en una combinación personalizada para tratar los registros coincidentes.

- `CrawlerConnection`: cadena UTF-8.

  Especifica el nombre de la conexión asociada con la tabla de catálogo utilizada.

- `TableSchema`: matriz de objetos Opción.

  La matriz de salida del esquema para un nodo determinado.

- `StagingTable`: cadena UTF-8.

  El nombre de la tabla provisional temporal que se utiliza al realizar MERGE o APPEND con upsert.

- `SelectedColumns`: matriz de objetos Opción.

  La lista de nombres de columnas que se utiliza para determinar un registro coincidente al realizar un MERGE o APPEND con upsert.

AmazonRedshiftAdvancedOption estructura

Especifica un valor opcional al conectarse al clúster de Redshift.

Campos

- `Key`: cadena UTF-8.

  La clave de la opción de conexión adicional.

- `Value`: cadena UTF-8.

  El valor de la opción de conexión adicional.

Estructura de opción

Especifica un valor de opción.

Campos

- `Value`: cadena UTF-8 que coincide con el Custom string pattern #40.

  Especifica el valor de la opción.

- `Label`: cadena UTF-8 que coincide con el Custom string pattern #40.

Especifica la etiqueta de la opción.

- `Description`: cadena UTF-8 que coincide con el Custom string pattern #40.

  Especifica la descripción de la opción.

CatalogSource estructura S3

Especifica un almacén de datos de Amazon S3 en el catálogo AWS Glue de datos.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del almacén de datos.
- `Database`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  La base de datos de lectura.
- `Table`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  La tabla de base de datos de lectura.
- `PartitionPredicate`: cadena UTF-8 que coincide con el Custom string pattern #40.

  Se eliminan las particiones que cumplen con este predicado. Los archivos comprendidos en el período de retención de estas particiones no se eliminan. Configurar en `""`, valor vacío de forma predeterminada.
- `AdditionalOptions`: un objeto S3 SourceAdditionalOptions.

  Especifica opciones de conexión adicionales.

SourceAdditionalOptions Estructura de S3

Especifica opciones de conexión adicionales para el almacén de datos de Amazon S3.

Campos

- `BoundedSize`: número (largo).

  Establece el límite superior del tamaño objetivo del conjunto de datos en bytes que se procesará.

- `BoundedFiles`: número (largo).

  Establece el límite superior del número objetivo de archivos que se procesarán.

CsvSource Estructura S3

Especifica un almacén de datos de valores separados por comas (CSV) almacenado en Amazon S3.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del almacén de datos.

- `Paths`: obligatorio: una matriz de cadenas UTF-8.

  Una lista de las rutas de Amazon S3 desde las que se leerá.

- `CompressionType`: cadena UTF-8 (valores válidos: `gzip="GZIP"` | `bzip2="BZIP2"`).

  Especifica la forma en que los datos se comprimen. Po lo general no es necesario si los datos tienen una extensión de archivo estándar. Los posibles valores son `"gzip"` y `"bzip"`).

- `Exclusions`: matriz de cadenas UTF-8.

  Una cadena que contiene una lista JSON de patrones glob de estilo Unix para excluir. Por ejemplo, "[\"**.pdf \"]" excluye todos los archivos PDF.

- `GroupSize`: cadena UTF-8 que coincide con el Custom string pattern #40.

  El tamaño del grupo de destino en bytes. El valor predeterminado se calcula en función del tamaño de los datos de entrada y el tamaño de su clúster. Cuando hay menos de 50 000 archivos de entrada, `"groupFiles"` debe establecerse en `"inPartition"` para que este valor surta efecto.

- `GroupFiles`: cadena UTF-8 que coincide con el Custom string pattern #40.

  La agrupación de archivos se habilita de forma predeterminada cuando la entrada contiene más de 50 000 archivos. Para habilitar las agrupaciones con menos de 50 000 archivos, establezca este parámetro en “inPartition”. Para deshabilitar las agrupaciones con más de 50 000 archivos, establezca este parámetro en `"none"`.

- `Recurse`: booleano.

Si se establece en verdadero, lee recursivamente archivos en todos los subdirectorios de las rutas especificadas.

- `MaxBand`: número (entero), cero como máximo.

  Esta opción controla la duración en milisegundos después de la que es probable que el listado de s3 sea coherente. Los archivos con marcas de tiempo de modificación que se encuentran dentro de los últimos milisegundos de MaxBand se rastrean especialmente cuando se utilizan JobBookmarks para tener en cuenta la coherencia eventual de Amazon S3. La mayoría de los usuarios no tienen que establecer esta opción. El valor predeterminado es 900 000 milisegundos, o 15 minutos.

- `MaxFilesInBand`: número (entero), cero como máximo.

  Esta opción especifica el número máximo de archivos que deben guardarse desde los últimos segundos de maxBand. Si se supera este número, los archivos adicionales se omiten y solo se procesarán en la siguiente ejecución del flujo de trabajo.

- `AdditionalOptions`: un objeto S3 DirectSourceAdditionalOptions.

  Especifica opciones de conexión adicionales.

- `Separator`: obligatorio: cadena de UTF-8 (valores válidos: `comma="COMMA"` | `ctrla="CTRLA"` | `pipe="PIPE"` | `semicolon="SEMICOLON"` | `tab="TAB"`).

  Especifica el carácter delimitador. El valor predeterminado es una coma: “,”; pero puede especificarse cualquier otro carácter.

- `Escaper`: cadena UTF-8 que coincide con el Custom string pattern #41.

  Especifica un carácter que se usar para aplicar escape. Esta opción solo se usa cuando se leen archivos CSV. El valor predeterminado es `none`. Si se habilita, el carácter que va inmediatamente después se usa tal cual, excepto un pequeño conjunto de escapes conocidos (`\n`, `\r`, `\t` y `\0`).

- `QuoteChar`: obligatorio: cadena UTF-8 (valores válidos: `quote="QUOTE"` | `quillemet="QUILLEMET"` | `single_quote="SINGLE_QUOTE"` | `disabled="DISABLED"`).

  Especifica el carácter que se usar para aplicar comillas. El carácter predeterminado es una comilla doble: `'"'`. Establezca esta opción en `-1` para desactivar las comillas por completo.

- `Multiline`: booleano.

  Un valor booleano que especifica si un solo registro puede abarcar varias líneas. Esto puede suceder cuando un campo contiene un carácter de nueva línea entre comillas. Debe configurar

esta opción en verdadero si un registro abarca varias líneas. El valor predeterminado es `False`, que permite una división de archivo más dinámica durante el análisis.

- `WithHeader`: booleano.

  Un valor booleano que especifica si la primera línea se debe tratar como un encabezado. El valor predeterminado es `False`.

- `WriteHeader`: booleano.

  Un valor booleano que especifica si se debe escribir el encabezado en la salida. El valor predeterminado es `True`.

- `SkipFirst`: booleano.

  Un valor booleano que especifica si se debe omitir la primera línea de datos. El valor predeterminado es `False`.

- `OptimizePerformance`: booleano.

  Un valor booleano que especifica si se debe utilizar el lector CSV SIMD avanzado junto con los formatos de memoria columnar con base en Apache Arrow. Solo disponible en AWS Glue la versión 3.0.

- `OutputSchemas`: matriz de objetos GlueSchema.

  Especifica el esquema de datos del origen de S3 con formato CSV.

Estructura DirectJDBCSource

Especifica la conexión de origen JDBC directa.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre de la conexión de origen de JDBC.

- `Database`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  La base de datos de la conexión de origen de JDBC.

- `Table`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  La tabla de la conexión de origen de JDBC.

- `ConnectionName`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de la conexión del origen de JDBC.
- `ConnectionType`: obligatorio: cadena de UTF-8 (valores válidos: `sqlserver` | `mysql` | `oracle` | `postgresql` | `redshift`).

  El tipo de conexión del origen de JDBC.
- `RedshiftTmpDir`: cadena UTF-8 que coincide con el Custom string pattern #40.

  El directorio temporal del origen de JDBC Redshift.

DirectSourceAdditionalOptions Estructura S3

Especifica opciones de conexión adicionales para el almacén de datos de Amazon S3.

Campos

- `BoundedSize`: número (largo).

  Establece el límite superior del tamaño objetivo del conjunto de datos en bytes que se procesará.
- `BoundedFiles`: número (largo).

  Establece el límite superior del número objetivo de archivos que se procesarán.
- `EnableSamplePath`: booleano.

  Establece la opción para habilitar una ruta de ejemplo.
- `SamplePath`: cadena UTF-8 que coincide con el Custom string pattern #40.

  Si está habilitado, especifica la ruta de ejemplo.

JsonSource Estructura S3

Especifica un almacén de datos JSON almacenado en Amazon S3.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del almacén de datos.

- `Paths`: obligatorio: una matriz de cadenas UTF-8.

  Una lista de las rutas de Amazon S3 desde las que se leerá.

- `CompressionType`: cadena UTF-8 (valores válidos: `gzip="GZIP"` | `bzip2="BZIP2"`).

  Especifica la forma en que los datos se comprimen. Po lo general no es necesario si los datos tienen una extensión de archivo estándar. Los posibles valores son `"gzip"` y `"bzip"`).

- `Exclusions`: matriz de cadenas UTF-8.

  Una cadena que contiene una lista JSON de patrones glob de estilo Unix para excluir. Por ejemplo, "[\"**.pdf \"]" excluye todos los archivos PDF.

- `GroupSize`: cadena UTF-8 que coincide con el Custom string pattern #40.

  El tamaño del grupo de destino en bytes. El valor predeterminado se calcula en función del tamaño de los datos de entrada y el tamaño de su clúster. Cuando hay menos de 50 000 archivos de entrada, `"groupFiles"` debe establecerse en `"inPartition"` para que este valor surta efecto.

- `GroupFiles`: cadena UTF-8 que coincide con el Custom string pattern #40.

  La agrupación de archivos se habilita de forma predeterminada cuando la entrada contiene más de 50 000 archivos. Para habilitar las agrupaciones con menos de 50 000 archivos, establezca este parámetro en “inPartition”. Para deshabilitar las agrupaciones con más de 50 000 archivos, establezca este parámetro en `"none"`.

- `Recurse`: booleano.

  Si se establece en verdadero, lee recursivamente archivos en todos los subdirectorios de las rutas especificadas.

- `MaxBand`: número (entero), cero como máximo.

  Esta opción controla la duración en milisegundos después de la que es probable que el listado de s3 sea coherente. Los archivos con marcas de tiempo de modificación que se encuentran dentro de los últimos milisegundos de MaxBand se rastrean especialmente cuando se utilizan JobBookmarks para tener en cuenta la coherencia eventual de Amazon S3. La mayoría de los usuarios no tienen que establecer esta opción. El valor predeterminado es 900 000 milisegundos, o 15 minutos.

- `MaxFilesInBand`: número (entero), cero como máximo.

Esta opción especifica el número máximo de archivos que deben guardarse desde los últimos segundos de maxBand. Si se supera este número, los archivos adicionales se omiten y solo se procesarán en la siguiente ejecución del flujo de trabajo.

- `AdditionalOptions`: un objeto S3 DirectSourceAdditionalOptions.

  Especifica opciones de conexión adicionales.

- `JsonPath`: cadena UTF-8 que coincide con el Custom string pattern #40.

  Una JsonPath cadena que define los datos de JSON.

- `Multiline`: booleano.

  Un valor booleano que especifica si un solo registro puede abarcar varias líneas. Esto puede suceder cuando un campo contiene un carácter de nueva línea entre comillas. Debe configurar esta opción en verdadero si un registro abarca varias líneas. El valor predeterminado es `False`, que permite una división de archivo más dinámica durante el análisis.

- `OutputSchemas`: matriz de objetos GlueSchema.

  Especifica el esquema de datos del origen de S3 con formato JSON.

ParquetSource Estructura S3

Especifica un almacén de datos de Apache Parquet almacenado en Amazon S3.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del almacén de datos.

- `Paths`: obligatorio: una matriz de cadenas UTF-8.

  Una lista de las rutas de Amazon S3 desde las que se leerá.

- `CompressionType`: cadena UTF-8 (valores válidos: `snappy="SNAPPY"` | `lzo="LZO"` | `gzip="GZIP"` | `uncompressed="UNCOMPRESSED"` | `none="NONE"`).

  Especifica la forma en que los datos se comprimen. Po lo general no es necesario si los datos tienen una extensión de archivo estándar. Los posibles valores son `"gzip"` y `"bzip"`).

- `Exclusions`: matriz de cadenas UTF-8.

Una cadena que contiene una lista JSON de patrones glob de estilo Unix para excluir. Por ejemplo, "[\"**.pdf \"]" excluye todos los archivos PDF.

- `GroupSize`: cadena UTF-8 que coincide con el Custom string pattern #40.

  El tamaño del grupo de destino en bytes. El valor predeterminado se calcula en función del tamaño de los datos de entrada y el tamaño de su clúster. Cuando hay menos de 50 000 archivos de entrada, `"groupFiles"` debe establecerse en `"inPartition"` para que este valor surta efecto.

- `GroupFiles`: cadena UTF-8 que coincide con el Custom string pattern #40.

  La agrupación de archivos se habilita de forma predeterminada cuando la entrada contiene más de 50 000 archivos. Para habilitar las agrupaciones con menos de 50 000 archivos, establezca este parámetro en “inPartition”. Para deshabilitar las agrupaciones con más de 50 000 archivos, establezca este parámetro en `"none"`.

- `Recurse`: booleano.

  Si se establece en verdadero, lee recursivamente archivos en todos los subdirectorios de las rutas especificadas.

- `MaxBand`: número (entero), cero como máximo.

  Esta opción controla la duración en milisegundos después de la que es probable que el listado de s3 sea coherente. Los archivos con marcas de tiempo de modificación que se encuentran dentro de los últimos milisegundos de MaxBand se rastrean especialmente cuando se utilizan JobBookmarks para tener en cuenta la coherencia eventual de Amazon S3. La mayoría de los usuarios no tienen que establecer esta opción. El valor predeterminado es 900 000 milisegundos, o 15 minutos.

- `MaxFilesInBand`: número (entero), cero como máximo.

  Esta opción especifica el número máximo de archivos que deben guardarse desde los últimos segundos de maxBand. Si se supera este número, los archivos adicionales se omiten y solo se procesarán en la siguiente ejecución del flujo de trabajo.

- `AdditionalOptions`: un objeto S3 DirectSourceAdditionalOptions.

  Especifica opciones de conexión adicionales.

- `OutputSchemas`: matriz de objetos GlueSchema.

  Especifica el esquema de datos del origen de S3 con formato Parquet.

Estructura de S3 DeltaSource

Especifica una fuente de datos de Delta Lake almacenada en Amazon S3.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del origen del Delta Lake.
- `Paths`: obligatorio: una matriz de cadenas UTF-8.

  Una lista de las rutas de Amazon S3 desde las que se leerá.
- `AdditionalDeltaOptions`: matriz de mapas de pares clave-valor.

  Cada clave es una cadena UTF-8 que coincide con el Custom string pattern #40.

  Cada valor es una cadena UTF-8 que coincide con el Custom string pattern #40.

  Especifica opciones de conexión adicionales.
- `AdditionalOptions`: un objeto S3 DirectSourceAdditionalOptions.

  Especifica las opciones adicionales para el conector.
- `OutputSchemas`: matriz de objetos GlueSchema.

  Especifica el esquema de datos para el origen de Delta Lake.

CatalogDeltaSource Estructura S3

Especifica una fuente de datos de Delta Lake que está registrada en el catálogo AWS Glue de datos. La fuente de datos debe almacenarse en Amazon S3.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del origen de datos de Delta Lake.
- `Database`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de la base de datos de lectura.

- `Table`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de la tabla en la base de datos de lectura.
- `AdditionalDeltaOptions`: matriz de mapas de pares clave-valor.

  Cada clave es una cadena UTF-8 que coincide con el Custom string pattern #40.

  Cada valor es una cadena UTF-8 que coincide con el Custom string pattern #40.

  Especifica opciones de conexión adicionales.
- `OutputSchemas`: matriz de objetos GlueSchema.

  Especifica el esquema de datos para el origen de Delta Lake.

CatalogDeltaSource estructura

Especifica una fuente de datos de Delta Lake que está registrada en el catálogo AWS Glue de datos.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del origen de datos de Delta Lake.
- `Database`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de la base de datos de lectura.
- `Table`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de la tabla en la base de datos de lectura.
- `AdditionalDeltaOptions`: matriz de mapas de pares clave-valor.

  Cada clave es una cadena UTF-8 que coincide con el Custom string pattern #40.

  Cada valor es una cadena UTF-8 que coincide con el Custom string pattern #40.

  Especifica opciones de conexión adicionales.
- `OutputSchemas`: matriz de objetos GlueSchema.

  Especifica el esquema de datos para el origen de Delta Lake.

HudiSource Estructura S3

Especifica una fuente de datos de Hudi almacenada en Amazon S3.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del origen de Hudi.
- `Paths`: obligatorio: una matriz de cadenas UTF-8.

  Una lista de las rutas de Amazon S3 desde las que se leerá.
- `AdditionalHudiOptions`: matriz de mapas de pares clave-valor.

  Cada clave es una cadena UTF-8 que coincide con el Custom string pattern #40.

  Cada valor es una cadena UTF-8 que coincide con el Custom string pattern #40.

  Especifica opciones de conexión adicionales.
- `AdditionalOptions`: un objeto S3 DirectSourceAdditionalOptions.

  Especifica las opciones adicionales para el conector.
- `OutputSchemas`: matriz de objetos GlueSchema.

  Especifica el esquema de datos para el origen de Hudi.

Estructura S3 CatalogHudiSource

Especifica una fuente de datos de Hudi que está registrada en el catálogo de AWS Glue datos. La fuente de datos de Hudi debe almacenarse en. Amazon S3

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del origen de datos de Hudi.
- `Database`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de la base de datos de lectura.

- `Table`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de la tabla en la base de datos de lectura.
- `AdditionalHudiOptions`: matriz de mapas de pares clave-valor.

  Cada clave es una cadena UTF-8 que coincide con el Custom string pattern #40.

  Cada valor es una cadena UTF-8 que coincide con el Custom string pattern #40.

  Especifica opciones de conexión adicionales.
- `OutputSchemas`: matriz de objetos GlueSchema.

  Especifica el esquema de datos para el origen de Hudi.

CatalogHudiSource estructura

Especifica una fuente de datos de Hudi que está registrada en el catálogo de AWS Glue datos.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del origen de datos de Hudi.
- `Database`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de la base de datos de lectura.
- `Table`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de la tabla en la base de datos de lectura.
- `AdditionalHudiOptions`: matriz de mapas de pares clave-valor.

  Cada clave es una cadena UTF-8 que coincide con el Custom string pattern #40.

  Cada valor es una cadena UTF-8 que coincide con el Custom string pattern #40.

  Especifica opciones de conexión adicionales.
- `OutputSchemas`: matriz de objetos GlueSchema.

  Especifica el esquema de datos para el origen de Hudi.

Estructura de DynamoDB CatalogSource

Especifica una fuente de datos de DynamoDB en AWS Glue el catálogo de datos.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del origen de datos.
- `Database`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de la base de datos de lectura.
- `Table`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de la tabla en la base de datos de lectura.

RelationalCatalogSource estructura

Especifica un origen de datos de base de datos relacional en AWS Glue Data Catalog.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del origen de datos.
- `Database`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de la base de datos de lectura.
- `Table`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de la tabla en la base de datos de lectura.

estructura JDBC ConnectorTarget

Especifica un destino de datos que escribe en Amazon S3 en el almacenamiento en columnas de Apache Parquet.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

El nombre del destino de datos.

- `Inputs`: Obligatorio: matriz de cadenas UTF-8, con una cadena como mínimo y una cadena como máximo.

  Los nodos que son entradas para el destino de datos.

- `ConnectionName`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de la conexión asociada al conector.

- `ConnectionTable`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #41.

  El nombre de la tabla en el destino de datos.

- `ConnectorName`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de un conector que se utilizará.

- `ConnectionType`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El tipo de conexión, como marketplace.jdbc o custom.jdbc, que designa una conexión a un destino de datos JDBC.

- `AdditionalOptions`: matriz de mapas de pares clave-valor.

  Cada clave es una cadena UTF-8 que coincide con el Custom string pattern #40.

  Cada valor es una cadena UTF-8 que coincide con el Custom string pattern #40.

  Opciones de conexión adicionales para el conector.

- `OutputSchemas`: matriz de objetos GlueSchema.

  Especifica el esquema de datos del destino de JDBC.

SparkConnectorTarget estructura

Especifica un destino que utiliza un conector de Apache Spark.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del destino de datos.

- `Inputs`: Obligatorio: matriz de cadenas UTF-8, con una cadena como mínimo y una cadena como máximo.

  Los nodos que son entradas para el destino de datos.
- `ConnectionName`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de una conexión de un conector de Apache Spark.
- `ConnectorName`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de un conector de Apache Spark.
- `ConnectionType`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El tipo de conexión, como marketplace.spark o custom.spark, que designa una conexión a un almacén de datos de Apache Spark.
- `AdditionalOptions`: matriz de mapas de pares clave-valor.

  Cada clave es una cadena UTF-8 que coincide con el Custom string pattern #40.

  Cada valor es una cadena UTF-8 que coincide con el Custom string pattern #40.

  Opciones de conexión adicionales para el conector.
- `OutputSchemas`: matriz de objetos GlueSchema.

  Especifica el esquema de datos del destino de spark personalizado.

BasicCatalogTarget estructura

Especifica un destino que utiliza una tabla AWS Glue de catálogo de datos.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del destino de datos.
- `Inputs`: Obligatorio: matriz de cadenas UTF-8, con una cadena como mínimo y una cadena como máximo.

  Los nodos que son entradas para el destino de datos.
- `Database`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

La base de datos que contiene la tabla que desea utilizar como destino. Esta base de datos ya debe existir en el Catálogo de datos.

- `Table`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  La tabla que define el esquema de los datos de salida. Esta tabla ya debe existir en el Catálogo de datos.

CatalogTarget Estructura de MySQL

Especifica un destino que utiliza MySQL.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del destino de datos.
- `Inputs`: Obligatorio: matriz de cadenas UTF-8, con una cadena como mínimo y una cadena como máximo.

  Los nodos que son entradas para el destino de datos.
- `Database`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de la base de datos de escritura.
- `Table`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de la tabla de la base de datos de escritura.

Estructura de PostgreSQL CatalogTarget

Especifica un destino que utiliza PostgreSQL.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del destino de datos.
- `Inputs`: Obligatorio: matriz de cadenas UTF-8, con una cadena como mínimo y una cadena como máximo.

Los nodos que son entradas para el destino de datos.

- `Database`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de la base de datos de escritura.

- `Table`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de la tabla de la base de datos de escritura.

Estructura de OracleSQL CatalogTarget

Especifica un destino que utiliza Oracle SQL.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del destino de datos.

- `Inputs`: Obligatorio: matriz de cadenas UTF-8, con una cadena como mínimo y una cadena como máximo.

  Los nodos que son entradas para el destino de datos.

- `Database`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de la base de datos de escritura.

- `Table`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de la tabla de la base de datos de escritura.

Estructura de Microsoft SQL ServerCatalogTarget

Especifica un destino que utiliza Microsoft SQL.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del destino de datos.

- `Inputs`: Obligatorio: matriz de cadenas UTF-8, con una cadena como mínimo y una cadena como máximo.

  Los nodos que son entradas para el destino de datos.
- `Database`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de la base de datos de escritura.
- `Table`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de la tabla de la base de datos de escritura.

RedshiftTarget estructura

Especifica un destino que utiliza Amazon Redshift.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del destino de datos.
- `Inputs`: Obligatorio: matriz de cadenas UTF-8, con una cadena como mínimo y una cadena como máximo.

  Los nodos que son entradas para el destino de datos.
- `Database`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de la base de datos de escritura.
- `Table`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de la tabla de la base de datos de escritura.
- `RedshiftTmpDir`: cadena UTF-8 que coincide con el Custom string pattern #40.

  La ruta de Amazon S3 donde se pueden almacenar datos temporales al copiar desde la base de datos.
- `TmpDirIAMRole`: cadena UTF-8 que coincide con el Custom string pattern #40.

  El rol de IAM con permisos.
- `UpsertRedshiftOptions`: un objeto UpsertRedshiftTargetOptions.

Conjunto de opciones para configurar una operación upsert al escribir en un destino de Redshift.

AmazonRedshiftTarget estructura

Especifica un destino de Amazon Redshift.

Campos

- `Name`: cadena UTF-8 que coincide con el Custom string pattern #43.

  Nombre del destino de Amazon Redshift.
- `Data`: un objeto AmazonRedshiftNodeData.

  Especifica los datos del nodo de destino de Amazon Redshift.
- `Inputs`: matriz de cadenas UTF-8, no menos de una cadena o más de una.

  Los nodos que son entradas para el destino de datos.

UpsertRedshiftTargetOptions estructura

Opciones para configurar una operación upsert al escribir en un destino de Redshift.

Campos

- `TableLocation`: cadena UTF-8 que coincide con el Custom string pattern #40.

  Ubicación física de la tabla de Redshift.
- `ConnectionName`: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de la conexión que se utilizará para escribir en Redshift.
- `UpsertKeys`: matriz de cadenas UTF-8.

  Claves utilizadas para determinar si se debe realizar una actualización o una inserción.

CatalogTarget estructura S3

Especifica un destino de datos que escribe en Amazon S3 mediante el catálogo AWS Glue de datos.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del destino de datos.
- `Inputs`: Obligatorio: matriz de cadenas UTF-8, con una cadena como mínimo y una cadena como máximo.

  Los nodos que son entradas para el destino de datos.
- `PartitionKeys`: matriz de cadenas UTF-8.

  Especifica la partición nativa mediante una secuencia de claves.
- `Table`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de la tabla de la base de datos de escritura.
- `Database`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de la base de datos de escritura.
- `SchemaChangePolicy`: un objeto CatalogSchemaChangePolicy.

  Una política que especifica los comportamientos de actualización del rastreador.

GlueParquetTarget Estructura de S3

Especifica un destino de datos que escribe en Amazon S3 en el almacenamiento en columnas de Apache Parquet.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del destino de datos.
- `Inputs`: Obligatorio: matriz de cadenas UTF-8, con una cadena como mínimo y una cadena como máximo.

  Los nodos que son entradas para el destino de datos.
- `PartitionKeys`: matriz de cadenas UTF-8.

  Especifica la partición nativa mediante una secuencia de claves.

- `Path`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  Una única ruta de Amazon S3 de escritura.
- `Compression`: cadena UTF-8 (valores válidos: `snappy="SNAPPY"` | `lzo="LZO"` | `gzip="GZIP"` | `uncompressed="UNCOMPRESSED"` | `none="NONE"`).

  Especifica la forma en que los datos se comprimen. Po lo general no es necesario si los datos tienen una extensión de archivo estándar. Los posibles valores son `"gzip"` y `"bzip"`).
- `SchemaChangePolicy`: un objeto DirectSchemaChangePolicy.

  Una política que especifica los comportamientos de actualización del rastreador.

CatalogSchemaChangePolicy estructura

Una política que especifica los comportamientos de actualización del rastreador.

Campos

- `EnableUpdateCatalog`: booleano.

  Si utilizar o no el comportamiento de actualización especificado cuando el rastreador encuentra un esquema cambiado.
- `UpdateBehavior`: cadena UTF-8 (valores válidos: `UPDATE_IN_DATABASE` | `LOG`).

  Comportamiento de actualización cuando el rastreador encuentra un esquema cambiado.

DirectTarget estructura S3

Especifica un destino de datos que escribe en Amazon S3.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del destino de datos.
- `Inputs`: Obligatorio: matriz de cadenas UTF-8, con una cadena como mínimo y una cadena como máximo.

  Los nodos que son entradas para el destino de datos.

- `PartitionKeys`: matriz de cadenas UTF-8.

  Especifica la partición nativa mediante una secuencia de claves.

- `Path`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  Una única ruta de Amazon S3 de escritura.

- `Compression`: cadena UTF-8 que coincide con el Custom string pattern #40.

  Especifica la forma en que los datos se comprimen. Po lo general no es necesario si los datos tienen una extensión de archivo estándar. Los posibles valores son `"gzip"` y `"bzip"`).

- `Format` – Obligatorio: cadena UTF-8 (valores válidos: `json="JSON"` | `csv="CSV"` | `avro="AVRO"` | `orc="ORC"` | `parquet="PARQUET"` | `hudi="HUDI"` | `delta="DELTA"`).

  Especifica el formato de salida de datos para el destino.

- `SchemaChangePolicy`: un objeto DirectSchemaChangePolicy.

  Una política que especifica los comportamientos de actualización del rastreador.

HudiCatalogTarget Estructura S3

Especifica un destino que escribe en una fuente de datos de Hudi del catálogo de AWS Glue datos.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del destino de datos.

- `Inputs`: Obligatorio: matriz de cadenas UTF-8, con una cadena como mínimo y una cadena como máximo.

  Los nodos que son entradas para el destino de datos.

- `PartitionKeys`: matriz de cadenas UTF-8.

  Especifica la partición nativa mediante una secuencia de claves.

- `Table`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de la tabla de la base de datos de escritura.

- `Database`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

El nombre de la base de datos de escritura.

- `AdditionalOptions` – Obligatorio: matriz de mapas de pares clave-valor.

  Cada clave es una cadena UTF-8 que coincide con el Custom string pattern #40.

  Cada valor es una cadena UTF-8 que coincide con el Custom string pattern #40.

  Especifica opciones de conexión adicionales para el conector.

- `SchemaChangePolicy`: un objeto CatalogSchemaChangePolicy.

  Una política que especifica los comportamientos de actualización del rastreador.

Estructura S3 HudiDirectTarget

Especifica un destino que escribe en una fuente de datos de Hudi. Amazon S3

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del destino de datos.

- `Inputs`: Obligatorio: matriz de cadenas UTF-8, con una cadena como mínimo y una cadena como máximo.

  Los nodos que son entradas para el destino de datos.

- `Path`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  La ruta de Amazon S3 del origen de datos de Hudi en la que desea escribir.

- `Compression`: obligatorio: cadena UTF-8 (valores válidos: `gzip="GZIP"` | `lzo="LZO"` | `uncompressed="UNCOMPRESSED"` | `snappy="SNAPPY"`).

  Especifica la forma en que los datos se comprimen. Po lo general no es necesario si los datos tienen una extensión de archivo estándar. Los posibles valores son `"gzip"` y `"bzip"`).

- `PartitionKeys`: matriz de cadenas UTF-8.

  Especifica la partición nativa mediante una secuencia de claves.

- `Format` – Obligatorio: cadena UTF-8 (valores válidos: `json="JSON"` | `csv="CSV"` | `avro="AVRO"` | `orc="ORC"` | `parquet="PARQUET"` | `hudi="HUDI"` | `delta="DELTA"`).

Especifica el formato de salida de datos para el destino.

- `AdditionalOptions` – Obligatorio: matriz de mapas de pares clave-valor.

  Cada clave es una cadena UTF-8 que coincide con el Custom string pattern #40.

  Cada valor es una cadena UTF-8 que coincide con el Custom string pattern #40.

  Especifica opciones de conexión adicionales para el conector.
- `SchemaChangePolicy`: un objeto DirectSchemaChangePolicy.

  Una política que especifica los comportamientos de actualización del rastreador.

Estructura S3 DeltaCatalogTarget

Especifica un destino que escribe en una fuente de datos de Delta Lake del catálogo AWS Glue de datos.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del destino de datos.
- `Inputs`: Obligatorio: matriz de cadenas UTF-8, con una cadena como mínimo y una cadena como máximo.

  Los nodos que son entradas para el destino de datos.
- `PartitionKeys`: matriz de cadenas UTF-8.

  Especifica la partición nativa mediante una secuencia de claves.
- `Table`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de la tabla de la base de datos de escritura.
- `Database`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de la base de datos de escritura.
- `AdditionalOptions`: matriz de mapas de pares clave-valor.

  Cada clave es una cadena UTF-8 que coincide con el Custom string pattern #40.

Cada valor es una cadena UTF-8 que coincide con el Custom string pattern #40.

Especifica opciones de conexión adicionales para el conector.

- `SchemaChangePolicy`: un objeto CatalogSchemaChangePolicy.

  Una política que especifica los comportamientos de actualización del rastreador.

DeltaDirectTarget Estructura S3

Especifica un destino que escribe en una fuente de datos de Delta Lake en Amazon S3.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del destino de datos.
- `Inputs`: Obligatorio: matriz de cadenas UTF-8, con una cadena como mínimo y una cadena como máximo.

  Los nodos que son entradas para el destino de datos.
- `PartitionKeys`: matriz de cadenas UTF-8.

  Especifica la partición nativa mediante una secuencia de claves.
- `Path`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  La ruta de Amazon S3 del origen de datos de Delta Lake en la que desea escribir.
- `Compression`: obligatorio: cadena UTF-8 (valores válidos: `uncompressed="UNCOMPRESSED"` | `snappy="SNAPPY"`).

  Especifica la forma en que los datos se comprimen. Po lo general no es necesario si los datos tienen una extensión de archivo estándar. Los posibles valores son `"gzip"` y `"bzip"`).
- `Format` – Obligatorio: cadena UTF-8 (valores válidos: `json="JSON"` | `csv="CSV"` | `avro="AVRO"` | `orc="ORC"` | `parquet="PARQUET"` | `hudi="HUDI"` | `delta="DELTA"`).

  Especifica el formato de salida de datos para el destino.
- `AdditionalOptions`: matriz de mapas de pares clave-valor.

  Cada clave es una cadena UTF-8 que coincide con el Custom string pattern #40.

Cada valor es una cadena UTF-8 que coincide con el Custom string pattern #40.

Especifica opciones de conexión adicionales para el conector.

- `SchemaChangePolicy`: un objeto DirectSchemaChangePolicy.

  Una política que especifica los comportamientos de actualización del rastreador.

DirectSchemaChangePolicy estructura

Una política que especifica los comportamientos de actualización del rastreador.

Campos

- `EnableUpdateCatalog`: booleano.

  Si utilizar o no el comportamiento de actualización especificado cuando el rastreador encuentra un esquema cambiado.

- `UpdateBehavior`: cadena UTF-8 (valores válidos: `UPDATE_IN_DATABASE` | `LOG`).

  Comportamiento de actualización cuando el rastreador encuentra un esquema cambiado.

- `Table`: cadena UTF-8 que coincide con el Custom string pattern #40.

  Especifica la tabla de la base de datos a la que se aplica la política de cambio de esquema.

- `Database`: cadena UTF-8 que coincide con el Custom string pattern #40.

  Especifica la base de datos a la que se aplica la política de cambio de esquema.

ApplyMapping estructura

Especifica una transformación que asigna claves de propiedad de datos en el origen de datos a claves de propiedad de datos en el destino de datos. Puede cambiar el nombre de las claves, modificar los tipos de datos de las claves y elegir las claves que desea descartar del conjunto de datos.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

El nombre del nodo de transformación.

- `Inputs`: obligatorio: matriz de cadenas UTF-8, con una cadena como mínimo y una cadena como máximo.

  Las entradas de datos identificadas por los nombres de sus nodos.

- `Mapping` (obligatorio): una matriz de objetos Correspondencia.

  Especifica la asignación de claves de propiedad de datos en el origen de datos a claves de propiedad de datos en el destino de datos.

Estructura de asignación

Especifica la asignación de claves de propiedad de datos.

Campos

- `ToKey`: cadena UTF-8 que coincide con el Custom string pattern #40.

  Después de aplicar la asignación, cuál debe ser el nombre de la columna. Puede ser igual que `FromPath`.

- `FromPath`: matriz de cadenas UTF-8.

  La tabla o columna que se va a modificar.

- `FromType`: cadena UTF-8 que coincide con el Custom string pattern #40.

  El tipo de datos que se van a modificar.

- `ToType`: cadena UTF-8 que coincide con el Custom string pattern #40.

  El tipo de datos al que se van a modificar los datos.

- `Dropped`: booleano.

  Si se establece en verdadero, se quita la columna.

- `Children`: matriz de objetos Correspondencia.

  Solo aplicable a estructuras de datos anidadas. Si desea cambiar la estructura principal, pero también una de las secundarias, puede rellenar esta estructura de datos. También es `Mapping`, pero su `FromPath` será el `FromPath` de la principal más el `FromPath` de esta estructura.

Para las partes secundarias, suponga que tiene la estructura:

```
{ "FromPath": "OuterStructure", "ToKey": "OuterStructure", "ToType":
"Struct", "Dropped": false, "Chidlren": [{ "FromPath": "inner", "ToKey":
"inner", "ToType": "Double", "Dropped": false, }] }
```

Puede especificar una `Mapping` que tiene este aspecto:

```
{ "FromPath": "OuterStructure", "ToKey": "OuterStructure", "ToType":
"Struct", "Dropped": false, "Chidlren": [{ "FromPath": "inner", "ToKey":
"inner", "ToType": "Double", "Dropped": false, }] }
```

SelectFields estructura

Especifica una transformación que elige las claves de propiedad de datos que desea conservar.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del nodo de transformación.

- `Inputs`: obligatorio: matriz de cadenas UTF-8, con una cadena como mínimo y una cadena como máximo.

  Las entradas de datos identificadas por los nombres de sus nodos.

- `Paths`: obligatorio: una matriz de cadenas UTF-8.

  Una ruta JSON a una variable de la estructura de datos.

DropFields estructura

Especifica una transformación que elige las claves de propiedad de datos que desea eliminar.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del nodo de transformación.

- `Inputs`: obligatorio: matriz de cadenas UTF-8, con una cadena como mínimo y una cadena como máximo.

  Las entradas de datos identificadas por los nombres de sus nodos.
- `Paths`: obligatorio: una matriz de cadenas UTF-8.

  Una ruta JSON a una variable de la estructura de datos.

RenameField estructura

Especifica una transformación que cambia el nombre de una única clave de propiedad de datos.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del nodo de transformación.
- `Inputs`: obligatorio: matriz de cadenas UTF-8, con una cadena como mínimo y una cadena como máximo.

  Las entradas de datos identificadas por los nombres de sus nodos.
- `SourcePath`: obligatorio: una matriz de cadenas UTF-8.

  Una ruta JSON a una variable de la estructura de datos de los datos de origen.
- `TargetPath`: obligatorio: una matriz de cadenas UTF-8.

  Una ruta JSON a una variable de la estructura de datos de los datos de destino.

Estructura Spigot

Especifica una transformación que escribe ejemplos de los datos en un bucket de Amazon S3.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del nodo de transformación.
- `Inputs`: obligatorio: matriz de cadenas UTF-8, con una cadena como mínimo y una cadena como máximo.

Las entradas de datos identificadas por los nombres de sus nodos.

- `Path`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  Una ruta en Amazon S3 donde la transformación escribe un subconjunto de registros del conjunto de datos en un archivo JSON en un bucket de Amazon S3.

- `Topk`: número (entero), 100 como máximo.

  Especifica un número de registros que se escribirán a partir del principio del conjunto de datos.

- `Prob`: número (doble), 1 como máximo.

  La probabilidad (un valor decimal con un valor máximo de 1) de seleccionar un registro determinado. Un valor 1 indica que cada fila leída del conjunto de datos debe incluirse en la salida de ejemplo.

Estructura Join

Especifica una transformación que une dos conjuntos de datos en uno mediante una frase de comparación en las claves de propiedad de datos especificadas. Puede utilizar combinaciones interna, externa, izquierda, derecha, semicombinación izquierda y anticombinación izquierda.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del nodo de transformación.

- `Inputs`: Obligatorio: matriz de cadenas UTF-8, con 2 cadenas como mínimo y 2 cadenas como máximo.

  Las entradas de datos identificadas por los nombres de sus nodos.

- `JoinType` Obligatorio: cadena UTF-8 (valores válidos: `equijoin="EQUIJOIN"` | `left="LEFT"` | `right="RIGHT"` | `outer="OUTER"` | `leftsemi="LEFT_SEMI"` | `leftanti="LEFT_ANTI"`).

  Especifica el tipo de unión que se va a realizar en los conjuntos de datos.

- `Columns`: Obligatorio: una matriz de objetos JoinColumn, con 2 estructuras como mínimo y 2 estructuras como máximo.

  Una lista de las dos columnas que se van a unir.

JoinColumn estructura

Especifica una columna que se va a unir.

Campos

- `From`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  La columna que se va a unir.
- `Keys`: obligatorio: una matriz de cadenas UTF-8.

  La clave de la columna que se va a unir.

SplitFields estructura

Especifica una transformación que divide las claves de propiedad de datos en dos `DynamicFrames`. La salida es una recopilación de `DynamicFrames`: uno con las claves de propiedad de datos seleccionadas y el otro con las claves de propiedad de datos restantes.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del nodo de transformación.
- `Inputs`: obligatorio: matriz de cadenas UTF-8, con una cadena como mínimo y una cadena como máximo.

  Las entradas de datos identificadas por los nombres de sus nodos.
- `Paths`: obligatorio: una matriz de cadenas UTF-8.

  Una ruta JSON a una variable de la estructura de datos.

SelectFromCollection estructura

Especifica una transformación que elige un `DynamicFrame` de una recopilación de `DynamicFrames`. El resultado es el `DynamicFrame` seleccionado

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

El nombre del nodo de transformación.

- `Inputs`: obligatorio: matriz de cadenas UTF-8, con una cadena como mínimo y una cadena como máximo.

  Las entradas de datos identificadas por los nombres de sus nodos.

- `Index` – Obligatorio: número (entero), cero como máximo.

  El índice del DynamicFrame que se va a seleccionar.

FillMissingValues estructura

Especifica una transformación que localiza registros en el conjunto de datos que tienen valores faltantes y agrega un nuevo campo con un valor determinado por imputación. El conjunto de datos de entrada se utiliza para formar al modelo de machine learning que determina cuál debe ser el valor que falta.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del nodo de transformación.

- `Inputs`: obligatorio: matriz de cadenas UTF-8, con una cadena como mínimo y una cadena como máximo.

  Las entradas de datos identificadas por los nombres de sus nodos.

- `ImputedPath`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  Una ruta JSON a una variable de la estructura de datos del conjunto de datos que se imputa.

- `FilledPath`: cadena UTF-8 que coincide con el Custom string pattern #40.

  Una ruta JSON a una variable de la estructura de datos del conjunto de datos que se rellena.

Estructura Filter

Especifica una transformación que divide un conjunto de datos en dos, en función de una condición de filtro.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del nodo de transformación.

- `Inputs`: obligatorio: matriz de cadenas UTF-8, con una cadena como mínimo y una cadena como máximo.

  Las entradas de datos identificadas por los nombres de sus nodos.

- `LogicalOperator`: obligatorio: cadena UTF-8 (valores válidos: `AND` | `OR`).

  El operador utilizado para filtrar filas mediante la comparación del valor de clave con un valor especificado.

- `Filters` (obligatorio): una matriz de objetos FilterExpression.

  Especifica una expresión de filtro.

FilterExpression estructura

Especifica una expresión de filtro.

Campos

- `Operation` – Obligatorio: cadena UTF-8 (valores válidos: `EQ` | `LT` | `GT` | `LTE` | `GTE` | `REGEX` | `ISNULL`).

  El tipo de operación que se va a realizar en la expresión.

- `Negated`: booleano.

  Si se va a negar la expresión.

- `Values` (obligatorio): una matriz de objetos FilterValue.

  Una lista de valores de filtro.

FilterValue estructura

Representa una única entrada en la lista de valores de una `FilterExpression`.

Campos

- `Type`: obligatorio: cadena UTF-8 (valores válidos: `COLUMNEXTRACTED` | `CONSTANT`).

  El tipo de valor de filtro.
- `Value`: obligatorio: una matriz de cadenas UTF-8.

  El valor que se va a asociar.

CustomCode estructura

Especifica una transformación que utiliza el código personalizado que proporciona el usuario para llevar a cabo la transformación de datos. La salida es una colección de DynamicFrames.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del nodo de transformación.
- `Inputs`: obligatorio: matriz de cadenas UTF-8, con una cadena como mínimo.

  Las entradas de datos identificadas por los nombres de sus nodos.
- `Code`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #35.

  El código personalizado que se utiliza para llevar a cabo la transformación de datos.
- `ClassName`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre definido para la clase de nodo de código personalizado.
- `OutputSchemas`: matriz de objetos GlueSchema.

  Especifica el esquema de datos para la transformación de código personalizada.

Estructura SparkSQL

Especifica una transformación en la que se ingresa una consulta SQL mediante la sintaxis de Spark SQL para transformar los datos. La salida es un único `DynamicFrame`.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del nodo de transformación.

- `Inputs`: obligatorio: matriz de cadenas UTF-8, con una cadena como mínimo.

  Las entradas de datos identificadas por los nombres de sus nodos. Puede asociar un nombre de tabla a cada nodo de entrada para utilizarlo en la consulta SQL. El nombre que elija debe cumplir las restricciones de nomenclatura de Spark SQL.

- `SqlQuery`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #42.

  Una consulta SQL que debe utilizar la sintaxis de Spark SQL y devolver un único conjunto de datos.

- `SqlAliases` (obligatorio): una matriz de objetos SqlAlias.

  Una lista de alias. Un alias permite especificar qué nombre se va a utilizar en SQL para una entrada determinada. Por ejemplo, tiene una fuente de datos llamada "»MyDataSource. Si especificas `From` como y `Alias` como MyDataSource SqlName, entonces en tu SQL puedes hacer lo siguiente:

  ```
  select * from SqlName
  ```

  y eso obtiene datos de MyDataSource.

- `OutputSchemas`: matriz de objetos GlueSchema.

  Especifica el esquema de datos para la transformación de SparkSQL.

SqlAlias estructura

Representa una única entrada en la lista de valores de `SqlAliases`.

Campos

- `From`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #39.

  Una tabla o columna de una tabla.

- `Alias`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #41.

Un nombre temporal dado a una tabla o a una columna de una tabla.

DropNullFields estructura

Especifica una transformación que elimina columnas del conjunto de datos si todos los valores de la columna son “nulos”. De forma predeterminada, AWS Glue Studio reconocerá los objetos nulos, pero algunos valores, como las cadenas vacías, las cadenas «nulas», los enteros -1 u otros marcadores de posición, como ceros, no se reconocen automáticamente como nulos.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del nodo de transformación.
- `Inputs`: obligatorio: matriz de cadenas UTF-8, con una cadena como mínimo y una cadena como máximo.

  Las entradas de datos identificadas por los nombres de sus nodos.
- `NullCheckBoxList`: un objeto NullCheckBoxList.

  Una estructura que representa si ciertos valores se reconocen como valores nulos para su eliminación.
- `NullTextList`: matriz de objetos NullValueField, con 50 estructuras como máximo.

  Estructura que especifica una lista de NullValueField estructuras que representan un valor nulo personalizado, como cero u otro valor, que se utiliza como marcador de posición nulo exclusivo del conjunto de datos.

  La transformación `DropNullFields` elimina los valores nulos personalizados solo si tanto el valor del marcador de posición nulo como el tipo de datos coinciden con los datos.

NullCheckBoxList estructura

Representa si ciertos valores se reconocen como valores nulos para su eliminación.

Campos

- `IsEmpty`: booleano.

Especifica que una cadena vacía se considera un valor nulo.

- `IsNullString`: booleano.

  Especifica que un valor que deletrea la palabra “nulo” se considera un valor nulo.

- `IsNegOne`: booleano.

  Especifica que un valor entero de -1 se considera un valor nulo.

NullValueField estructura

Representa un valor nulo personalizado, como ceros u otro valor que se utiliza como marcador de posición nulo exclusivo del conjunto de datos.

Campos

- `Value`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El valor del marcador de posición nulo.

- `Datatype`: obligatorio: objeto Tipo de datos.

  El tipo de datos del valor.

Estructura Datatype

Una estructura que representa el tipo de datos del valor.

Campos

- `Id`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #39.

  El tipo de datos del valor.

- `Label`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #39.

  Una etiqueta asignada al tipo de datos.

Estructura Merge

Especifica una transformación que fusiona un `DynamicFrame` con una instancia provisional de `DynamicFrame` en función de las claves principales especificadas para identificar registros. Los registros duplicados (registros con las mismas claves principales) no se eliminan.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del nodo de transformación.

- `Inputs`: Obligatorio: matriz de cadenas UTF-8, con 2 cadenas como mínimo y 2 cadenas como máximo.

  Las entradas de datos identificadas por los nombres de sus nodos.

- `Source`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #39.

  El `DynamicFrame` de origen que se fusionará con un `DynamicFrame` de instancia provisional.

- `PrimaryKeys`: obligatorio: una matriz de cadenas UTF-8.

  La lista de campos de clave principal para hacer coincidir los registros de los marcos dinámicos de origen y provisionales.

Estructura Union

Especifica una transformación que combina las filas de dos o más conjuntos de datos en un único resultado.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del nodo de transformación.

- `Inputs`: Obligatorio: matriz de cadenas UTF-8, con 2 cadenas como mínimo y 2 cadenas como máximo.

  Las entradas del ID de nodo a la transformación.

- `UnionType`: obligatorio: cadena UTF-8 (valores válidos: `ALL` | `DISTINCT`).

Indica el tipo de transformación de combinación.

Especifique `ALL` unir todas las filas de las fuentes de datos a las resultantes DynamicFrame. La combinación resultante no elimina las filas duplicadas.

Especifique `DISTINCT` si desea eliminar las filas duplicadas del resultado DynamicFrame.

Estructura PIIDetection

Especifica una transformación que identifica, elimina o enmascara datos PII.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del nodo de transformación.
- `Inputs`: obligatorio: matriz de cadenas UTF-8, con una cadena como mínimo y una cadena como máximo.

  Las entradas del ID de nodo a la transformación.
- `PiiType`: obligatorio: cadena UTF-8 (valores válidos: `RowAudit` | `RowMasking` | `ColumnAudit` | `ColumnMasking`).

  Indica el tipo de transformación PIIDDetection.
- `EntityTypesToDetect`: obligatorio: una matriz de cadenas UTF-8.

  Indica los tipos de entidades que la transformación PIIDProtection identificará como datos PII.

  Las entidades de tipo PII incluyen: PERSON_NAME, DATE, USA_SNN, EMAIL, USA_ITIN, USA_PASSPORT_NUMBER, PHONE_NUMBER, BANK_ACCOUNT, IP_ADDRESS, MAC_ADDRESS, USA_CPT_CODE, USA_HCPCS_CODE, USA_NATIONAL_DRUG_CODE, USA_MEDICARE_BENEFICIARY_IDENTIFIER, USA_HEALTH_INSURANCE_CLAIM_NUMBER,CREDIT_CARD,USA_NATIONAL_PROVIDER_IDENTIFIE
- `OutputColumnName`: cadena UTF-8 que coincide con el Custom string pattern #40.

  Indica el nombre de columna de salida que contendrá cualquier tipo de entidad detectado en esa fila.
- `SampleFraction`: número (doble), 1 como máximo.

Indica la fracción de los datos que se van a muestrear al buscar entidades PII.

- `ThresholdFraction`: número (doble), 1 como máximo.

  Indica la fracción de los datos que deben cumplirse para que una columna se identifique como datos de PII.

- `MaskValue`: cadena UTF-8, de 256 bytes de largo como máximo, que coincide con Custom string pattern #37.

  Indica el valor que sustituirá a la entidad detectada.

Estructura Aggregate

Especifica una transformación que agrupa las filas según los campos elegidos y calcula el valor agregado mediante una función especificada.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del nodo de transformación.

- `Inputs`: obligatorio: matriz de cadenas UTF-8, con una cadena como mínimo y una cadena como máximo.

  Especifica los campos y filas que se utilizarán como entradas para la transformación agregada.

- `Groups`: obligatorio: una matriz de cadenas UTF-8.

  Especifica los campos para agrupar.

- `Aggs`: obligatorio: una matriz de objetos AggregateOperation, con 1 estructura como mínimo y 30 estructuras como máximo.

  Especifica las funciones agregadas que se van a realizar en campos especificados.

DropDuplicates estructura

Especifica una transformación que elimina las filas de datos repetidos de un conjunto de datos.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del nodo de transformación.

- `Inputs`: obligatorio: matriz de cadenas UTF-8, con una cadena como mínimo y una cadena como máximo.

  Las entradas de datos identificadas por los nombres de sus nodos.

- `Columns`: matriz de cadenas UTF-8.

  Nombre de las columnas que se van a fusionar o eliminar si se repiten.

GovernedCatalogTarget estructura

Especifica un destino de datos que escribe en Amazon S3 mediante el catálogo AWS Glue de datos.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del destino de datos.

- `Inputs`: Obligatorio: matriz de cadenas UTF-8, con una cadena como mínimo y una cadena como máximo.

  Los nodos que son entradas para el destino de datos.

- `PartitionKeys`: matriz de cadenas UTF-8.

  Especifica la partición nativa mediante una secuencia de claves.

- `Table`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de la tabla de la base de datos de escritura.

- `Database`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El nombre de la base de datos de escritura.

- `SchemaChangePolicy`: un objeto CatalogSchemaChangePolicy.

  Una política que especifica los comportamientos de actualización del catálogo gobernado.

GovernedCatalogSource estructura

Especifica el banco de datos del catálogo AWS Glue de datos regulado.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del almacén de datos.
- `Database`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  La base de datos de lectura.
- `Table`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  La tabla de base de datos de lectura.
- `PartitionPredicate`: cadena UTF-8 que coincide con el Custom string pattern #40.

  Se eliminan las particiones que cumplen con este predicado. Los archivos comprendidos en el período de retención de estas particiones no se eliminan. Configurar en `""`, valor vacío de forma predeterminada.
- `AdditionalOptions`: un objeto S3 SourceAdditionalOptions.

  Especifica opciones de conexión adicionales.

AggregateOperation estructura

Especifica el conjunto de parámetros necesarios para llevar a cabo la agregación en la transformación de agregación.

Campos

- `Column`: obligatorio: una matriz de cadenas UTF-8.

  Especifica la columna del conjunto de datos en la que se aplicará la función de agregación.
- `AggFunc`: obligatorio: cadena UTF-8 (valores válidos: `avg` | `countDistinct` | `count` | `first` | `last` | `kurtosis` | `max` | `min` | `skewness` | `stddev_samp` | `stddev_pop` | `sum` | `sumDistinct` | `var_samp` | `var_pop`).

  Especifica la función de agregación que se aplicará.

Las posibles funciones de agregación incluyen: avg countDistinct, count, first, last, kurtosis, max, min, skewness, stddev_samp, stddev_pop, sum, sumDistinct, var_samp, var_pop

GlueSchema estructura

Especifica un esquema definido por el usuario cuando un esquema no puede determinarse mediante AWS Glue.

Campos

- `Columns`: matriz de objetos GlueStudioSchemaColumn.

  Especifica las definiciones de columnas que componen un AWS Glue esquema.

GlueStudioSchemaColumn estructura

Especifica una sola columna en una definición AWS Glue de esquema.

Campos

- `Name` – Obligatorio: cadena UTF-8, de 1024 bytes de largo como máximo, que coincide con Single-line string pattern.

  El nombre de la columna en el esquema de AWS Glue Studio.
- `Type`: cadena UTF-8 con un máximo de 131072 bytes de largo, que coincide con el Single-line string pattern.

  El tipo de colmena de esta columna en el esquema de AWS Glue Studio.

GlueStudioColumn estructura

Especifica una sola columna en AWS Glue Studio.

Campos

- `Key`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #41.

  La clave de la columna en AWS Glue Studio.

- `FullPath`: obligatorio: una matriz de cadenas UTF-8.

  La URL completa de la columna en AWS Glue Studio.

- `Type` – Obligatorio: matriz UTF-8 (valores válidos: `array="ARRAY"` | `bigint="BIGINT"` | `bigint array="BIGINT_ARRAY"` | `binary="BINARY"` | `binary array="BINARY_ARRAY"` | `boolean="BOOLEAN"` | `boolean array="BOOLEAN_ARRAY"` | `byte="BYTE"` | `byte array="BYTE_ARRAY"` | `char="CHAR"` | `char array="CHAR_ARRAY"` | `choice="CHOICE"` | `choice array="CHOICE_ARRAY"` | `date="DATE"` | `date array="DATE_ARRAY"` | `decimal="DECIMAL"` | `decimal array="DECIMAL_ARRAY"` | `double="DOUBLE"` | `double array="DOUBLE_ARRAY"` | `enum="ENUM"` | `enum array="ENUM_ARRAY"` | `float="FLOAT"` | `float array="FLOAT_ARRAY"` | `int="INT"` | `int array="INT_ARRAY"` | `interval="INTERVAL"` | `interval array="INTERVAL_ARRAY"` | `long="LONG"` | `long array="LONG_ARRAY"` | `object="OBJECT"` | `short="SHORT"` | `short array="SHORT_ARRAY"` | `smallint="SMALLINT"` | `smallint array="SMALLINT_ARRAY"` | `string="STRING"` | `string array="STRING_ARRAY"` | `timestamp="TIMESTAMP"` | `timestamp array="TIMESTAMP_ARRAY"` | `tinyint="TINYINT"` | `tinyint array="TINYINT_ARRAY"` | `varchar="VARCHAR"` | `varchar array="VARCHAR_ARRAY"` | `null="NULL"` | `unknown="UNKNOWN"` | `unknown array="UNKNOWN_ARRAY"`).

  El tipo de columna en AWS Glue Studio.

- `Children`: un conjunto de estructuras.

  Los hijos de la columna principal en AWS Glue Studio.

DynamicTransform estructura

Especifica el conjunto de parámetros necesarios para hacer la transformación dinámica.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  Especifica el nombre de la transformación dinámica.

- `TransformName`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  Especifica el nombre de la transformación dinámica tal como aparece en el editor visual de AWS Glue Studio.

- `Inputs`: obligatorio: matriz de cadenas UTF-8, con una cadena como mínimo y una cadena como máximo.

  Especifica las entradas necesarias para la transformación dinámica.
- `Parameters`: matriz de objetos TransformConfigParameter.

  Especifica los parámetros de la transformación dinámica.
- `FunctionName`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  Especifica el nombre de la función de la transformación dinámica.
- `Path`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  Especifica la ruta de los archivos de origen y de configuración de la transformación dinámica.
- `Version`: cadena UTF-8 que coincide con el Custom string pattern #40.

  Este campo no se utiliza y quedará en desuso en la versión futura.
- `OutputSchemas`: matriz de objetos GlueSchema.

  Especifica el esquema de datos para la transformación dinámica.

TransformConfigParameter estructura

Especifica los parámetros de la transformación dinámica en el archivo de configuración.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  Especifica el nombre del parámetro de la transformación dinámica en el archivo de configuración.
- `Type` – Obligatorio: cadena UTF-8 (valores válidos: `str="STR"` | `int="INT"` | `float="FLOAT"` | `complex="COMPLEX"` | `bool="BOOL"` | `list="LIST"` | `null="NULL"`).

  Especifica el tipo de parámetro de la transformación dinámica en el archivo de configuración.
- `ValidationRule`: cadena UTF-8 que coincide con el Custom string pattern #40.

  Especifica la regla de validación de la transformación dinámica en el archivo de configuración.
- `ValidationMessage`: cadena UTF-8 que coincide con el Custom string pattern #40.

  Especifica el mensaje de validación de la transformación dinámica en el archivo de configuración.

- `Value`: matriz de cadenas UTF-8.

  Especifica el valor del parámetro de la transformación dinámica en el archivo de configuración.

- `ListType`: cadena UTF-8 (valores válidos: `str="STR"` | `int="INT"` | `float="FLOAT"` | `complex="COMPLEX"` | `bool="BOOL"` | `list="LIST"` | `null="NULL"`).

  Especifica el tipo de lista del parámetro de la transformación dinámica en el archivo de configuración.

- `IsOptional`: booleano.

  Especifica si es opcional o no el parámetro de la transformación dinámica en el archivo de configuración.

EvaluateDataQuality estructura

Especifica los criterios de evaluación de la calidad de los datos.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre de la evaluación de la calidad de los datos.

- `Inputs`: obligatorio: matriz de cadenas UTF-8, con una cadena como mínimo y una cadena como máximo.

  Las entradas de la evaluación de la calidad de los datos.

- `Ruleset`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 65 536 bytes de largo como máximo, que coincide con Custom string pattern #38.

  El conjunto de reglas para la evaluación de la calidad de los datos.

- `Output`: cadena UTF-8 (valores válidos: `PrimaryInput` | `EvaluationResults`).

  La salida de la evaluación de la calidad de los datos.

- `PublishingOptions`: un objeto DQ ResultsPublishingOptions.

  Opciones para configurar cómo se publican los resultados.

- `StopJobOnFailureOptions`: un objeto DQ StopJobOnFailureOptions.

Opciones para configurar la forma en que se detendrá el trabajo si se produce un error en la evaluación de la calidad de los datos.

estructura DQ ResultsPublishingOptions

Opciones para configurar cómo se publican los resultados de la evaluación de la calidad de los datos.

Campos

- `EvaluationContext`: cadena UTF-8 que coincide con el Custom string pattern #39.

  El contexto de la evaluación.
- `ResultsS3Prefix`: cadena UTF-8 que coincide con el Custom string pattern #40.

  El prefijo de Amazon S3 se antepuso a los resultados.
- `CloudWatchMetricsEnabled`: booleano.

  Habilite las métricas de los resultados de la calidad de datos.
- `ResultsPublishingEnabled`: booleano.

  Habilite la publicación de los resultados de la calidad de datos.

Estructura DQ StopJobOnFailureOptions

Opciones para configurar la forma en que se detendrá el trabajo si se produce un error en la evaluación de la calidad de los datos.

Campos

- `StopJobOnFailureTiming`: cadena UTF-8 (valores válidos: `Immediate` | `AfterDataLoad`).

  Cuándo detener el trabajo si se produce un error en la evaluación de la calidad de los datos. Las opciones son inmediatas o. AfterDataLoad

EvaluateDataQualityMultiFrame estructura

Especifica los criterios de evaluación de la calidad de los datos.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre de la evaluación de la calidad de los datos.

- `Inputs`: obligatorio: matriz de cadenas UTF-8, con una cadena como mínimo.

  Las entradas de la evaluación de la calidad de los datos. La primera entrada en esta lista es el origen de datos principal.

- `AdditionalDataSources`: matriz de mapas de pares clave-valor.

  Cada clave es una cadena UTF-8 que coincide con el Custom string pattern #43.

  Cada valor es una cadena UTF-8 que coincide con el Custom string pattern #40.

  Los alias de todas los orígenes de datos excepto las principales.

- `Ruleset`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 65 536 bytes de largo como máximo, que coincide con Custom string pattern #38.

  El conjunto de reglas para la evaluación de la calidad de los datos.

- `PublishingOptions`: un objeto DQ ResultsPublishingOptions.

  Opciones para configurar cómo se publican los resultados.

- `AdditionalOptions`: matriz de mapas de pares clave-valor.

  Cada clave es una cadena UTF-8 (valores válidos: `performanceTuning.caching="CacheOption"` | `observations.scope="ObservationsOption"`).

  Cada valor es una cadena UTF-8.

  Opciones para configurar el comportamiento de la transformación en tiempo de ejecución.

- `StopJobOnFailureOptions`: un objeto DQ StopJobOnFailureOptions.

  Opciones para configurar la forma en que se detendrá el trabajo si se produce un error en la evaluación de la calidad de los datos.

Estructura de receta

Un nodo de AWS Glue Studio que usa una AWS Glue DataBrew receta en los AWS Glue trabajos.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del nodo de AWS Glue Studio.
- `Inputs`: obligatorio: matriz de cadenas UTF-8, con una cadena como mínimo y una cadena como máximo.

  Los nodos de entrada al nodo de la receta, identificados mediante el ID.
- `RecipeReference`: obligatorio: objeto RecipeReference.

  Una referencia a la DataBrew receta utilizada por el nodo.

RecipeReference estructura

Referencia a una AWS Glue DataBrew receta.

Campos

- `RecipeArn`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El ARN de la DataBrew receta.
- `RecipeVersion` - Obligatorio: cadena UTF-8, con no menos de 1 byte de largo o más de 16.

  El RecipeVersion de la DataBrew receta.

SnowflakeNodeData estructura

Especifica la configuración de los nodos de Snowflake en AWS Glue Studio.

Campos

- `SourceType`: cadena UTF-8 que coincide con el Custom string pattern #39.

  Especifica cómo se especifican los datos recuperados. Valores válidos: `"table"`, `"query"`.

- `Connection`: un objeto Opción.

  Especifica una conexión del catálogo AWS Glue de datos a un punto final de Snowflake.

- `Schema`: cadena UTF-8.

  Especifica un esquema de base de datos de Snowflake para que lo utilice el nodo.

- `Table`: cadena UTF-8.

  Especifica una tabla de Snowflake para que lo utilice el nodo.

- `Database`: cadena UTF-8.

  Especifica una base de datos de Snowflake para que lo utilice el nodo.

- `TempDir`: cadena UTF-8 que coincide con el Custom string pattern #40.

  En la actualidad no se utiliza.

- `IamRole`: un objeto Opción.

  En la actualidad no se utiliza.

- `AdditionalOptions`: matriz de mapas de pares clave-valor.

  Cada clave es una cadena UTF-8 que coincide con el Custom string pattern #40.

  Cada valor es una cadena UTF-8 que coincide con el Custom string pattern #40.

  Especifica opciones adicionales que se pasan al conector de Snowflake. Si las opciones se especifican en otra parte de este nodo, esto tendrá prioridad.

- `SampleQuery`: cadena UTF-8.

  Una cadena de SQL que se utiliza para recuperar datos con el sourcetype `query`.

- `PreAction`: cadena UTF-8.

  Una cadena de SQL que se ejecuta antes de que el conector Snowflake lleve a cabo sus acciones estándar.

- `PostAction`: cadena UTF-8.

  Una cadena SQL que se ejecuta después de que el conector Snowflake lleve a cabo sus acciones estándar.

- `Action`: cadena UTF-8.

Especifica qué acción se debe realizar al escribir en una tabla con datos preexistentes. Valores válidos: `append`, `merge`, `truncate`, `drop`.

- `Upsert`: booleano.

  Se utiliza cuando Action es `append`. Especifica el comportamiento de la resolución cuando ya existe una fila. Si es verdadero, se actualizarán las filas preexistentes. Si es falso, se insertarán esas filas.

- `MergeAction`: cadena UTF-8 que coincide con el Custom string pattern #39.

  Especifica una acción de combinación. Valores válidos: `simple`, `custom`. Si es simple, el comportamiento de combinación se define mediante `MergeWhenMatched` y `MergeWhenNotMatched`. Si es personalizado, se define mediante `MergeClause`.

- `MergeWhenMatched`: cadena UTF-8 que coincide con el Custom string pattern #39.

  Especifica cómo resolver los registros que coinciden con datos preexistentes al combinarlos. Valores válidos: `update`, `delete`.

- `MergeWhenNotMatched`: cadena UTF-8 que coincide con el Custom string pattern #39.

  Especifica cómo procesar los registros que no coinciden con los datos preexistentes al combinarlos. Valores válidos: `insert`, `none`.

- `MergeClause`: cadena UTF-8.

  Una instrucción de SQL que especifica un comportamiento de combinación personalizado.

- `StagingTable`: cadena UTF-8.

  El nombre de una tabla de preparación que se utiliza al realizar acciones `merge` o upsert `append`. Los datos se escriben en esta tabla y, a continuación, se mueven a `table` mediante una acción posterior generada.

- `SelectedColumns`: matriz de objetos Opción.

  Especifica las columnas combinadas para identificar un registro al detectar coincidencias de combinaciones y alteraciones. Una lista de estructuras con claves `value`, `label` y `description`. Cada estructura describe una columna.

- `AutoPushdown`: booleano.

Especifica si está habilitada la función automática de consultas pushdown. Si la función pushdown está habilitada, cuando se ejecuta una consulta en Spark, si parte de la consulta se puede "enviar" al servidor de Snowflake, se empuja hacia abajo. Esto mejora el rendimiento de algunas consultas.

- `TableSchema`: matriz de objetos Opción.

  Define manualmente el esquema de destino del nodo. Una lista de estructuras con claves `value`, `label` y `description`. Cada estructura describe una columna.

SnowflakeSource estructura

Especifica un origen de datos de Snowflake.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del origen de datos de Snowflake.

- `Data`: obligatorio: objeto SnowflakeNodeData.

  Configuración del origen de datos de Snowflake.

- `OutputSchemas`: matriz de objetos GlueSchema.

  Especifica los esquemas definidos por el usuario para los datos de salida.

SnowflakeTarget estructura

Especifica un destino de Snowflake.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre del origen de Snowflake.

- `Data`: obligatorio: objeto SnowflakeNodeData.

  Especifica los datos del nodo de destino de Snowflake.

- `Inputs`: matriz de cadenas UTF-8, no menos de una cadena o más de una.

Los nodos que son entradas para el destino de datos.

ConnectorDataSource estructura

Especifica una fuente generada con opciones de conexión estándar.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre de este nodo de origen.

- `ConnectionType`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El`ConnectionType`, tal como se proporciona a la AWS Glue biblioteca subyacente. Este tipo de nodo admite los siguientes tipos de conexión:

  - `opensearch`
  - `azuresql`
  - `azurecosmos`
  - `bigquery`
  - `saphana`
  - `teradata`
  - `vertica`

- `Data` – Obligatorio: matriz de mapas de pares clave-valor.

  Cada clave es una cadena UTF-8.

  Cada valor es una cadena UTF-8.

  Un mapa que especifica opciones de conexión para el nodo. Puede encontrar las opciones de conexión estándar para el tipo de conexión correspondiente en la sección Parámetros de conexión de la AWS Glue documentación.

- `OutputSchemas`: matriz de objetos GlueSchema.

  Especifica el esquema de datos para este origen.

ConnectorDataTarget estructura

Especifica un destino generado con opciones de conexión estándar.

Campos

- `Name`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #43.

  El nombre de este nodo de destino.
- `ConnectionType`: obligatorio: cadena UTF-8 que coincide con el Custom string pattern #40.

  El`ConnectionType`, tal como se proporciona a la AWS Glue biblioteca subyacente. Este tipo de nodo admite los siguientes tipos de conexión:
  - `opensearch`
  - `azuresql`
  - `azurecosmos`
  - `bigquery`
  - `saphana`
  - `teradata`
  - `vertica`
- `Data` – Obligatorio: matriz de mapas de pares clave-valor.

  Cada clave es una cadena UTF-8.

  Cada valor es una cadena UTF-8.

  Un mapa que especifica opciones de conexión para el nodo. Puede encontrar las opciones de conexión estándar para el tipo de conexión correspondiente en la sección Parámetros de conexión de la AWS Glue documentación.
- `Inputs`: matriz de cadenas UTF-8, no menos de una cadena o más de una.

  Los nodos que son entradas para el destino de datos.

API de trabajos

La API de trabajos describe los tipos de datos de trabajos y contiene las API para utilizar trabajos, ejecuciones de trabajos y desencadenadores en AWS Glue.

Temas

- Jobs
- Ejecuciones de trabajo
- Desencadenadores

Jobs

La API de Jobs describe los tipos de datos y la API relacionados con la creación, actualización, eliminación o visualización de trabajos en AWS Glue.

Tipos de datos

- Estructura de trabajo
- ExecutionProperty estructura
- NotificationProperty estructura
- JobCommand estructura
- ConnectionsList estructura
- JobUpdate estructura
- SourceControlDetails estructura

Estructura de trabajo

Especifica una definición de flujo de trabajo.

Campos

- `Name`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre que se asigna a esta definición de flujo de trabajo.
- `JobMode`: cadena UTF-8 (valores válidos: `SCRIPT=""` | `VISUAL=""` | `NOTEBOOK=""`).

  Un modo que describe cómo se creó un trabajo. Los valores válidos son:
  - `SCRIPT`- El trabajo se creó con el editor de scripts de AWS Glue Studio.
  - `VISUAL`- El trabajo se creó con el editor visual de AWS Glue Studio.

  - `NOTEBOOK`: El trabajo se creó con un cuaderno de sesiones interactivo.

  Cuando el campo `JobMode` no aparece o es nulo, se asigna `SCRIPT` como valor predeterminado.
- `Description`: cadena de descripción de un máximo de 2048 bytes de largo, que coincide con el URI address multi-line string pattern.

  Una descripción del trabajo.
- `LogUri`: cadena UTF-8.

  Este campo se reserva para un uso ulterior.
- `Role`: cadena UTF-8.

  El nombre o nombre de recurso de Amazon (ARN) del rol de IAM asociado a este trabajo.
- `CreatedOn`: marca temporal.

  La fecha y hora en que se creó esta definición de flujo de trabajo.
- `LastModifiedOn`: marca temporal.

  La última vez en que se modificó esta definición de flujo de trabajo.
- `ExecutionProperty`: un objeto ExecutionProperty.

  Un `ExecutionProperty` que especifica el número máximo de ejecuciones simultáneas permitidas para este trabajo.
- `Command`: un objeto JobCommand.

  El `JobCommand` que ejecuta este trabajo.
- `DefaultArguments`: matriz de mapas de pares clave-valor.

  Cada clave es una cadena UTF-8.

  Cada valor es una cadena UTF-8.

  Los argumentos predeterminados para cada ejecución de este trabajo, que se especifican como pares nombre-valor.

  Aquí puede especificar los argumentos que consume su propio script de ejecución de tareas, así como los argumentos que consume él AWS Glue mismo.

Es posible que se registren los argumentos del trabajo. No utilice secretos con formato de texto no cifrado como argumentos. Recupera los secretos de una AWS Glue Conexión AWS Secrets Manager u otro mecanismo de gestión de secretos si pretendes mantenerlos en el Job.

Para obtener información acerca de cómo especificar y utilizar sus propios argumentos de trabajo, consulte Llamadas a las API de AWS Glue en Python en la guía para desarrolladores.

Para obtener información sobre los argumentos que puede proporcionar a este campo al configurar los trabajos de Spark, consulte el tema Parámetros especiales utilizados por AWS Glue en la guía para desarrolladores.

Para obtener información sobre los argumentos que puede proporcionar a este campo al configurar los trabajos de Ray, consulte Utilizar parámetros de trabajo utilizados en trabajos de Ray en la guía para desarrolladores.

- `NonOverridableArguments`: matriz de mapas de pares clave-valor.

  Cada clave es una cadena UTF-8.

  Cada valor es una cadena UTF-8.

  Los argumentos para este trabajo que no se anulan cuando se proporcionan argumentos de trabajo en una ejecución de trabajos, se especifican como pares nombre-valor.

- `Connections`: un objeto ConnectionsList.

  Las conexiones que se utilizan para este flujo de trabajo.

- `MaxRetries`: número (entero).

  El número máximo de veces que se puede volver a intentar este trabajo después de un JobRun error.

- `AllocatedCapacity`: número (entero).

  Este campo está obsoleto. En su lugar, use `MaxCapacity`.

  El número de unidades de procesamiento de AWS Glue datos (DPU) asignadas a las ejecuciones de este trabajo. Puede asignar un mínimo de 2 DPU; el valor predeterminado es 10. Una DPU es una medida relativa de la potencia de procesamiento que consta de 4 vCPU de capacidad de cómputo y 16 GB de memoria. Para obtener más información, consulte la página de precios de AWS Glue.

- `Timeout`: número (entero), como mínimo 1.

  El tiempo de espera del flujo de trabajo en minutos. Es el tiempo máximo que una ejecución de trabajo puede consumir recursos antes de que se termine y cambie al estado `TIMEOUT`. El valor predeterminado es 2 880 minutos (48 horas) para los trabajos por lotes.

  Los trabajos de streaming deben tener valores de tiempo de espera inferiores a 7 días o 10 080 minutos. Si el valor se deja en blanco, el trabajo se reiniciará al cabo de 7 días si no se ha configurado un período de mantenimiento. Si ha configurado un período de mantenimiento, se reiniciará durante el período de mantenimiento a los 7 días.

- `MaxCapacity`: número (doble).

  Para los trabajos de Glue versión 1.0 o anteriores, utilizando el tipo de trabajador estándar, el número de unidades de procesamiento de AWS Glue datos (DPU) que se pueden asignar cuando se ejecuta este trabajo. Una DPU es una medida relativa de la potencia de procesamiento que consta de 4 vCPU de capacidad de cómputo y 16 GB de memoria. Para obtener más información, consulte la página de precios de AWS Glue.

  En los trabajos de Glue versión 2.0 o posterior, no puede especificar un `Maximum capacity`. En su lugar, debe especificar un `Worker type` y el `Number of workers`.

  No establezca `MaxCapacity` si utiliza `WorkerType` y `NumberOfWorkers`.

  El valor que se puede asignar a `MaxCapacity` depende de si se está ejecutando un trabajo de shell de Python, un trabajo de ETL de Apache Spark o un trabajo de ETL de streaming de Apache Spark:

  - Cuando especifica un trabajo de shell de Python (`JobCommand.Name`="pythonshell"), puede asignar 0,0625 o 1 DPU. El valor predeterminado es 0,0625 DPU.
  - Cuando especifica un trabajo ETL de Apache Spark (`JobCommand.Name`="glueetl") o un trabajo de ETL de streaming de Apache Spark (`JobCommand.Name`="gluestreaming"), puede asignar de 2 a 100 DPU. El valor predeterminado es 10 DPU. Este tipo de trabajo no puede tener una asignación de DPU fraccionaria.

- `WorkerType`: cadena UTF-8 (valores válidos: `Standard=""` | `G.1X=""` | `G.2X=""` | `G.025X=""` | `G.4X=""` | `G.8X=""` | `Z.2X=""`).

  El tipo de proceso de trabajo predefinido que se asigna cuando se ejecuta un trabajo. Acepta un valor de G.1X, G.2X, G.4X, G.8X o G.025X para los trabajos de Spark. Acepta el valor Z.2X para los trabajos Ray.

- Para el tipo de trabajador `G.1X`, cada trabajador se asocia a 1 DPU (4 vCPU, 16 GB de memoria) con un disco de 84 GB (aproximadamente 34 GB libres) y proporciona 1 ejecutor por trabajador. Recomendamos este tipo de trabajador para cargas de trabajo como transformaciones de datos, uniones y consultas, ya que ofrece una forma escalable y rentable de ejecutar la mayoría de los trabajos.
- Para el tipo de trabajador `G.2X`, cada trabajador se asocia a 2 DPU (8 vCPU, 32 GB de memoria) con un disco de 128 GB (aproximadamente 77 GB libres) y proporciona 1 ejecutor por trabajador. Recomendamos este tipo de trabajador para cargas de trabajo como transformaciones de datos, uniones y consultas, ya que ofrece una forma escalable y rentable de ejecutar la mayoría de los trabajos.
- Para el tipo de trabajador `G.4X`, cada trabajador se asocia a 4 DPU (16 vCPU, 64 GB de memoria) con un disco de 256 GB (aproximadamente 235 GB libres) y proporciona 1 ejecutor por trabajador. Recomendamos este tipo de trabajador para los trabajos cuyas cargas de trabajo contengan las transformaciones, agregaciones, uniones y consultas más exigentes. Este tipo de trabajador solo está disponible para los trabajos de Spark ETL de la AWS Glue versión 3.0 o posteriores en AWS las siguientes regiones: EE.UU. Este (Ohio), EE.UU. Este (Norte de Virginia), EE.UU. Oeste (Oregón), Asia Pacífico (Singapur), Asia Pacífico (Sídney), Asia Pacífico (Tokio), Canadá (Central), Europa (Fráncfort), Europa (Irlanda) y Europa (Estocolmo).
- Para el tipo de trabajador `G.8X`, cada trabajador se asocia a 8 DPU (32 GB vCPU, 128 GB de memoria) con un disco de 512 GB (aproximadamente 487 GB libres) y proporciona 1 ejecutor por trabajador. Recomendamos este tipo de trabajador para los trabajos cuyas cargas de trabajo contengan las transformaciones, agregaciones, uniones y consultas más exigentes. Este tipo de trabajador solo está disponible para los trabajos ETL de Spark de la AWS Glue versión 3.0 o posteriores, en las mismas AWS regiones compatibles con el tipo de `G.4X` trabajador.
- Para el tipo de trabajador `G.025X`, cada trabajador se asigna a 0,25 DPU (2 vCPU, 4 GB de memoria) con un disco de 84 GB (aproximadamente 34 GB libres) y proporciona 1 ejecutor por trabajador. Le recomendamos este tipo de proceso de trabajo para trabajos de streaming de bajo volumen. Este tipo de trabajador solo está disponible para los trabajos de streaming de la AWS Glue versión 3.0.
- Para el tipo de trabajador `Z.2X`, cada trabajador se asigna a 2 M-DPU (8 vCPU, 64 GB de memoria) con un disco de 128 GB (aproximadamente 120 GB libres) y proporciona hasta 8 trabajadores de Ray en función del escalador automático.

- `NumberOfWorkers`: número (entero).

El número de procesos de trabajo de `workerType` definido que se asignan cuando se ejecuta un trabajo.

- `SecurityConfiguration`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la estructura `SecurityConfiguration` que se va a utilizar con este trabajo.

- `NotificationProperty`: un objeto NotificationProperty.

  Especifica las propiedades de configuración de una notificación de flujo de trabajo.

- `Running`: booleano.

  Este campo se reserva para un uso ulterior.

- `GlueVersion`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Custom string pattern #20.

  En los trabajos de Spark, `GlueVersion` determina las versiones de Apache Spark y Python que AWS Glue están disponibles en un trabajo. La versión de Python indica la versión admitida para trabajos de tipo Spark.

  Los trabajos de Ray se deben configurar `GlueVersion` en `4.0` o superior. Sin embargo, las versiones de Ray, Python y bibliotecas adicionales que están disponibles en el trabajo de Ray están determinadas por el parámetro `Runtime` del comando del trabajo.

  Para obtener más información sobre las AWS Glue versiones disponibles y las correspondientes versiones de Spark y Python, consulta la versión de Glue en la guía para desarrolladores.

  Los trabajos que se crean sin especificar una versión de Glue se establecen de forma predeterminada en Glue 0.9.

- `CodeGenConfigurationNodes`: matriz de mapas de pares clave-valor.

  Cada clave es una cadena UTF-8 que coincide con el Custom string pattern #39.

  Cada valor es un objeto A CodeGenConfigurationNode.

  La representación de un gráfico acíclico dirigido en el que se basa tanto el componente visual de Glue Studio como la generación de código de Glue Studio.

- `ExecutionClass`: cadena UTF-8 de 16 bytes de largo como máximo (valores válidos: `FLEX=""` | `STANDARD=""`).

Indica si el trabajo se ejecuta con una clase de ejecución estándar o flexible. La clase de ejecución estándar es ideal para cargas de trabajo urgentes que requieren un inicio rápido de los trabajos y recursos dedicados.

La clase de ejecución flexible es adecuada para trabajos no urgentes cuyos momentos de inicio y finalización pueden variar.

Solo se `glueetl` podrán configurar `ExecutionClass` los trabajos con la AWS Glue versión 3.0 o superior y el tipo de comando`FLEX`. La clase de ejecución flexible está disponible para los trabajos de Spark.

- `SourceControlDetails`: un objeto SourceControlDetails.

  Los detalles de una configuración de control de código de origen para un trabajo, que permite la sincronización de los artefactos del trabajo hacia o desde un repositorio remoto.

- `MaintenanceWindow`: cadena UTF-8 que coincide con el Custom string pattern #30.

  Este campo especifica un día de la semana y una hora para el período de mantenimiento de los trabajos de streaming. AWS Glue realiza actividades de mantenimiento periódicamente. Durante estos períodos de mantenimiento, AWS Glue tendrás que reiniciar tus trabajos de streaming.

  AWS Glue reiniciará el trabajo en un plazo de 3 horas a partir del período de mantenimiento especificado. Por ejemplo, si configura el período de mantenimiento para el lunes a las 10:00 h GMT, sus trabajos se reiniciarán entre las 10:00 h GMT y las 13:00 h GMT.

- `ProfileName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de un perfil de AWS Glue uso asociado al trabajo.

ExecutionProperty estructura

Propiedad de ejecución de un flujo de trabajo.

Campos

- `MaxConcurrentRuns`: número (entero).

El número máximo de ejecuciones simultáneas que están permitidas para el flujo de trabajo. El valor predeterminado de es 1. Se produce un error cuando se llega a este umbral. El valor máximo que puede especificar se controla mediante un límite de servicio.

NotificationProperty estructura

Especifica las propiedades de configuración de una notificación.

Campos

- `NotifyDelayAfter`: número (entero), como mínimo 1.

  Después de que comience una ejecución de flujo de trabajo, el número de minutos que se debe esperar antes de enviar una notificación de retraso de ejecución de un flujo de trabajo.

JobCommand estructura

Especifica el código ejecutado cuando se ejecuta un trabajo.

Campos

- `Name`: cadena UTF-8.

  El nombre del comando de trabajo. Para un trabajo de ETL de Apache Spark, este debe ser `glueetl`. Para un trabajo de shell de Python, debe ser `pythonshell`. Para un trabajo de ETL de streaming Apache Spark, este debe ser `gluestreaming`. Para un trabajo de Ray, esto debe ser `glueray`.
- `ScriptLocation`: cadena UTF-8 de 400 000 bytes de largo como máximo.

  Especifica la ruta de Amazon Simple Storage Service (Amazon S3) a un script que ejecuta un trabajo.
- `PythonVersion`: cadena UTF-8 que coincide con el Custom string pattern #21.

  La versión de Python que se utiliza para ejecutar un trabajo de shell de Python. Los valores permitidos son 2 o 3.
- `Runtime`: cadena UTF-8, de 64 bytes de largo como máximo, que coincide con Custom string pattern #29.

En los trabajos de Ray, el tiempo de ejecución se utiliza para especificar las versiones de Ray, Python y bibliotecas adicionales disponibles en su entorno. Este campo no se usa en otros tipos de trabajos. Para conocer los valores de los entornos de tiempo de ejecución compatibles, consulte los entornos de tiempo de ejecución de Ray compatibles en la Guía para AWS Glue desarrolladores.

ConnectionsList estructura

Especifica las conexiones que utiliza un flujo de trabajo.

Campos

- `Connections`: matriz de cadenas UTF-8.

  Lista de conexiones que utiliza el flujo de trabajo.

JobUpdate estructura

Especifica la información que se utiliza para actualizar una definición de trabajo existente. Esta información sobrescribe por completo la definición del trabajo anterior.

Campos

- `JobMode`: cadena UTF-8 (valores válidos: `SCRIPT=""` | `VISUAL=""` | `NOTEBOOK=""`).

  Un modo que describe cómo se creó un trabajo. Los valores válidos son:
  - `SCRIPT`- El trabajo se creó con el editor de scripts de AWS Glue Studio.
  - `VISUAL`- El trabajo se creó con el editor visual de AWS Glue Studio.
  - `NOTEBOOK`: El trabajo se creó con un cuaderno de sesiones interactivo.

  Cuando el campo `JobMode` no aparece o es nulo, se asigna `SCRIPT` como valor predeterminado.
- `Description`: cadena de descripción de un máximo de 2048 bytes de largo, que coincide con el URI address multi-line string pattern.

  Descripción del flujo de trabajo que se va a definir.
- `LogUri`: cadena UTF-8.

  Este campo se reserva para un uso ulterior.

- `Role`: cadena UTF-8.

  El nombre o nombre de recurso de Amazon (ARN) del rol de IAM asociado a este trabajo (obligatorio).
- `ExecutionProperty`: un objeto ExecutionProperty.

  Un `ExecutionProperty` que especifica el número máximo de ejecuciones simultáneas permitidas para este trabajo.
- `Command`: un objeto JobCommand.

  El `JobCommand` que ejecuta este trabajo (obligatorio).
- `DefaultArguments`: matriz de mapas de pares clave-valor.

  Cada clave es una cadena UTF-8.

  Cada valor es una cadena UTF-8.

  Los argumentos predeterminados para cada ejecución de este trabajo, que se especifican como pares nombre-valor.

  Aquí puede especificar los argumentos que consume su propio script de ejecución de tareas, así como los argumentos que consume él AWS Glue mismo.

  Es posible que se registren los argumentos del trabajo. No utilice secretos con formato de texto no cifrado como argumentos. Recupera los secretos de una AWS Glue Conexión AWS Secrets Manager u otro mecanismo de gestión de secretos si pretendes mantenerlos en el Job.

  Para obtener información acerca de cómo especificar y utilizar sus propios argumentos de trabajo, consulte Llamadas a las API de AWS Glue en Python en la guía para desarrolladores.

  Para obtener información sobre los argumentos que puede proporcionar a este campo al configurar los trabajos de Spark, consulte el tema Parámetros especiales utilizados por AWS Glue en la guía para desarrolladores.

  Para obtener información sobre los argumentos que puede proporcionar a este campo al configurar los trabajos de Ray, consulte Utilizar parámetros de trabajo utilizados en trabajos de Ray en la guía para desarrolladores.
- `NonOverridableArguments`: matriz de mapas de pares clave-valor.

  Cada clave es una cadena UTF-8.

Cada valor es una cadena UTF-8.

Los argumentos para este trabajo que no se anulan cuando se proporcionan argumentos de trabajo en una ejecución de trabajos, se especifican como pares nombre-valor.

- `Connections`: un objeto ConnectionsList.

  Las conexiones que se utilizan para este flujo de trabajo.

- `MaxRetries`: número (entero).

  Número máximo de reintentos permitidos para esta tarea si se genera un error.

- `AllocatedCapacity`: número (entero).

  Este campo está obsoleto. En su lugar, use `MaxCapacity`.

  El número de unidades de procesamiento de AWS Glue datos (DPU) que se van a asignar a esta tarea. Puede asignar un mínimo de 2 DPU; el valor predeterminado es 10. Una DPU es una medida relativa de la potencia de procesamiento que consta de 4 vCPU de capacidad de cómputo y 16 GB de memoria. Para obtener más información, consulte la página de precios de AWS Glue.

- `Timeout`: número (entero), como mínimo 1.

  El tiempo de espera del flujo de trabajo en minutos. Es el tiempo máximo que una ejecución de trabajo puede consumir recursos antes de que se termine y cambie al estado `TIMEOUT`. El valor predeterminado es 2 880 minutos (48 horas) para los trabajos por lotes.

  Los trabajos de streaming deben tener valores de tiempo de espera inferiores a 7 días o 10 080 minutos. Si el valor se deja en blanco, el trabajo se reiniciará al cabo de 7 días si no se ha configurado un período de mantenimiento. Si ha configurado un período de mantenimiento, se reiniciará durante el período de mantenimiento a los 7 días.

- `MaxCapacity`: número (doble).

  Para los trabajos de Glue versión 1.0 o anteriores, utilizando el tipo de trabajador estándar, el número de unidades de procesamiento de AWS Glue datos (DPU) que se pueden asignar cuando se ejecuta este trabajo. Una DPU es una medida relativa de la potencia de procesamiento que consta de 4 vCPU de capacidad de cómputo y 16 GB de memoria. Para obtener más información, consulte la página de precios de AWS Glue.

  En los trabajos de Glue versión 2.0+, no puede especificar un `Maximum capacity`. En su lugar, debe especificar un `Worker type` y el `Number of workers`.

No establezca `MaxCapacity` si utiliza `WorkerType` y `NumberOfWorkers`.

El valor que se puede asignar a `MaxCapacity` depende de si se está ejecutando un trabajo de shell de Python, un trabajo de ETL de Apache Spark o un trabajo de ETL de streaming de Apache Spark:

- Cuando especifica un trabajo de shell de Python (`JobCommand.Name`="pythonshell"), puede asignar 0,0625 o 1 DPU. El valor predeterminado es 0,0625 DPU.
- Cuando especifica un trabajo ETL de Apache Spark (`JobCommand.Name`="glueetl") o un trabajo de ETL de streaming de Apache Spark (`JobCommand.Name`="gluestreaming"), puede asignar de 2 a 100 DPU. El valor predeterminado es 10 DPU. Este tipo de trabajo no puede tener una asignación de DPU fraccionaria.

- `WorkerType`: cadena UTF-8 (valores válidos: `Standard=""` | `G.1X=""` | `G.2X=""` | `G.025X=""` | `G.4X=""` | `G.8X=""` | `Z.2X=""`).

  El tipo de proceso de trabajo predefinido que se asigna cuando se ejecuta un trabajo. Acepta un valor de G.1X, G.2X, G.4X, G.8X o G.025X para los trabajos de Spark. Acepta el valor Z.2X para los trabajos Ray.

  - Para el tipo de trabajador `G.1X`, cada trabajador se asocia a 1 DPU (4 vCPU, 16 GB de memoria) con un disco de 84 GB (aproximadamente 34 GB libres) y proporciona 1 ejecutor por trabajador. Recomendamos este tipo de trabajador para cargas de trabajo como transformaciones de datos, uniones y consultas, ya que ofrece una forma escalable y rentable de ejecutar la mayoría de los trabajos.
  - Para el tipo de trabajador `G.2X`, cada trabajador se asocia a 2 DPU (8 vCPU, 32 GB de memoria) con un disco de 128 GB (aproximadamente 77 GB libres) y proporciona 1 ejecutor por trabajador. Recomendamos este tipo de trabajador para cargas de trabajo como transformaciones de datos, uniones y consultas, ya que ofrece una forma escalable y rentable de ejecutar la mayoría de los trabajos.
  - Para el tipo de trabajador `G.4X`, cada trabajador se asocia a 4 DPU (16 vCPU, 64 GB de memoria) con un disco de 256 GB (aproximadamente 235 GB libres) y proporciona 1 ejecutor por trabajador. Recomendamos este tipo de trabajador para los trabajos cuyas cargas de trabajo contengan las transformaciones, agregaciones, uniones y consultas más exigentes. Este tipo de trabajador solo está disponible para los trabajos de Spark ETL de la AWS Glue versión 3.0 o posteriores en AWS las siguientes regiones: EE.UU. Este (Ohio), EE.UU. Este (Norte de Virginia), EE.UU. Oeste (Oregón), Asia Pacífico (Singapur), Asia Pacífico (Sídney), Asia Pacífico (Tokio), Canadá (Central), Europa (Fráncfort), Europa (Irlanda) y Europa (Estocolmo).

  - Para el tipo de trabajador `G.8X`, cada trabajador se asocia a 8 DPU (32 GB vCPU, 128 GB de memoria) con un disco de 512 GB (aproximadamente 487 GB libres) y proporciona 1 ejecutor por trabajador. Recomendamos este tipo de trabajador para los trabajos cuyas cargas de trabajo contengan las transformaciones, agregaciones, uniones y consultas más exigentes. Este tipo de trabajador solo está disponible para los trabajos ETL de Spark de la AWS Glue versión 3.0 o posteriores, en las mismas AWS regiones compatibles con el tipo de `G.4X` trabajador.
  - Para el tipo de trabajador `G.025X`, cada trabajador se asigna a 0,25 DPU (2 vCPU, 4 GB de memoria) con un disco de 84 GB (aproximadamente 34 GB libres) y proporciona 1 ejecutor por trabajador. Le recomendamos este tipo de proceso de trabajo para trabajos de streaming de bajo volumen. Este tipo de trabajador solo está disponible para los trabajos de streaming de la AWS Glue versión 3.0.
  - Para el tipo de trabajador `Z.2X`, cada trabajador se asigna a 2 M-DPU (8 vCPU, 64 GB de memoria) con un disco de 128 GB (aproximadamente 120 GB libres) y proporciona hasta 8 trabajadores de Ray en función del escalador automático.
- `NumberOfWorkers`: número (entero).

  El número de procesos de trabajo de `workerType` definido que se asignan cuando se ejecuta un trabajo.
- `SecurityConfiguration`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la estructura `SecurityConfiguration` que se va a utilizar con este trabajo.
- `NotificationProperty`: un objeto NotificationProperty.

  Especifica las propiedades de configuración de una notificación de trabajo.
- `GlueVersion`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Custom string pattern #20.

  En los trabajos de Spark, `GlueVersion` determina las versiones de Apache Spark y Python que AWS Glue están disponibles en un trabajo. La versión de Python indica la versión admitida para trabajos de tipo Spark.

  Los trabajos de Ray se deben configurar `GlueVersion` en `4.0` o superior. Sin embargo, las versiones de Ray, Python y bibliotecas adicionales que están disponibles en el trabajo de Ray están determinadas por el parámetro `Runtime` del comando del trabajo.

Para obtener más información sobre las AWS Glue versiones disponibles y las correspondientes versiones de Spark y Python, consulta la versión de Glue en la guía para desarrolladores.

Los trabajos que se crean sin especificar una versión de Glue se establecen de forma predeterminada en Glue 0.9.

- `CodeGenConfigurationNodes`: matriz de mapas de pares clave-valor.

  Cada clave es una cadena UTF-8 que coincide con el Custom string pattern #39.

  Cada valor es un objeto A CodeGenConfigurationNode.

  La representación de un gráfico acíclico dirigido en el que se basa tanto el componente visual de Glue Studio como la generación de código de Glue Studio.

- `ExecutionClass`: cadena UTF-8 de 16 bytes de largo como máximo (valores válidos: `FLEX=""` | `STANDARD=""`).

  Indica si el trabajo se ejecuta con una clase de ejecución estándar o flexible. La clase de ejecución estándar es ideal para cargas de trabajo urgentes que requieren un inicio rápido de los trabajos y recursos dedicados.

  La clase de ejecución flexible es adecuada para trabajos no urgentes cuyos momentos de inicio y finalización pueden variar.

  Solo se `glueetl` podrán configurar `ExecutionClass` los trabajos con la AWS Glue versión 3.0 o superior y el tipo de comando`FLEX`. La clase de ejecución flexible está disponible para los trabajos de Spark.

- `SourceControlDetails`: un objeto SourceControlDetails.

  Los detalles de una configuración de control de código de origen para un trabajo, que permite la sincronización de los artefactos del trabajo hacia o desde un repositorio remoto.

- `MaintenanceWindow`: cadena UTF-8 que coincide con el Custom string pattern #30.

  Este campo especifica un día de la semana y una hora para el período de mantenimiento de los trabajos de streaming. AWS Glue realiza actividades de mantenimiento periódicamente. Durante estos períodos de mantenimiento, AWS Glue tendrás que reiniciar tus trabajos de streaming.

AWS Glue reiniciará el trabajo en un plazo de 3 horas a partir del período de mantenimiento especificado. Por ejemplo, si configura el período de mantenimiento para el lunes a las 10:00 h GMT, sus trabajos se reiniciarán entre las 10:00 h GMT y las 13:00 h GMT.

- `ProfileName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de un perfil de AWS Glue uso asociado al trabajo.

SourceControlDetails estructura

Los detalles de una configuración de control de código de origen para un trabajo, que permite la sincronización de los artefactos del trabajo hacia o desde un repositorio remoto.

Campos

- `Provider`: cadena UTF-8.

  El proveedor del repositorio remoto.
- `Repository`: cadena UTF-8, con 1 byte de largo como mínimo y 512 bytes de largo como máximo.

  Nombre del repositorio remoto que contiene los artefactos del trabajo.
- `Owner`: cadena UTF-8, con 1 byte de largo como mínimo y 512 bytes de largo como máximo.

  Propietario del repositorio remoto que contiene los artefactos del trabajo.
- `Branch`: cadena UTF-8, con 1 byte de largo como mínimo y 512 bytes de largo como máximo.

  Una rama opcional en el repositorio remoto.
- `Folder`: cadena UTF-8, con 1 byte de largo como mínimo y 512 bytes de largo como máximo.

  Una carpeta opcional en el repositorio remoto.
- `LastCommitId`: cadena UTF-8, con 1 byte de largo como mínimo y 512 bytes de largo como máximo.

  El último identificador de confirmación de una confirmación en el repositorio remoto.
- `LastSyncTimestamp`: cadena UTF-8, con 1 byte de largo como mínimo y 512 bytes de largo como máximo.

La fecha y hora en que se realizó la última sincronización del trabajo.

- `AuthStrategy`: cadena UTF-8.

  El tipo de autenticación, que puede ser un token de autenticación almacenado en AWS Secrets Manager o un token de acceso personal.

- `AuthToken`: cadena UTF-8, con 1 byte de largo como mínimo y 512 bytes de largo como máximo.

  El valor de un token de autorización.

Operaciones

- CreateJob acción (Python: create_job)
- UpdateJob acción (Python: update_job)
- GetJob acción (Python: get_job)
- GetJobs acción (Python: get_jobs)
- DeleteJob acción (Python: delete_job)
- ListJobs acción (Python: list_jobs)
- BatchGetJobs acción (Python: batch_get_jobs)

CreateJob acción (Python: create_job)

Crea una nueva definición de flujo de trabajo.

Solicitud

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre que se asigna a esta definición de flujo de trabajo. Debe ser único en la cuenta de .

- `JobMode`: cadena UTF-8 (valores válidos: `SCRIPT=""` | `VISUAL=""` | `NOTEBOOK=""`).

  Un modo que describe cómo se creó un trabajo. Los valores válidos son:

  - `SCRIPT`- El trabajo se creó con el editor de scripts de AWS Glue Studio.
  - `VISUAL`- El trabajo se creó con el editor visual de AWS Glue Studio.
  - `NOTEBOOK`: El trabajo se creó con un cuaderno de sesiones interactivo.

Cuando el campo `JobMode` no aparece o es nulo, se asigna `SCRIPT` como valor predeterminado.

- `Description`: cadena de descripción de un máximo de 2048 bytes de largo, que coincide con el URI address multi-line string pattern.

  Descripción del flujo de trabajo que se va a definir.

- `LogUri`: cadena UTF-8.

  Este campo se reserva para un uso ulterior.

- `Role` – Obligatorio: cadena UTF-8.

  El nombre o nombre de recurso de Amazon (ARN) del rol de IAM asociado a este trabajo.

- `ExecutionProperty`: un objeto ExecutionProperty.

  Un `ExecutionProperty` que especifica el número máximo de ejecuciones simultáneas permitidas para este trabajo.

- `Command`: obligatorio: objeto JobCommand.

  El `JobCommand` que ejecuta este trabajo.

- `DefaultArguments`: matriz de mapas de pares clave-valor.

  Cada clave es una cadena UTF-8.

  Cada valor es una cadena UTF-8.

  Los argumentos predeterminados para cada ejecución de este trabajo, que se especifican como pares nombre-valor.

  Aquí puede especificar los argumentos que consume su propio script de ejecución de tareas, así como los argumentos que consume él AWS Glue mismo.

  Es posible que se registren los argumentos del trabajo. No utilice secretos con formato de texto no cifrado como argumentos. Recupera los secretos de una AWS Glue Conexión AWS Secrets Manager u otro mecanismo de gestión de secretos si pretendes mantenerlos en el Job.

  Para obtener información acerca de cómo especificar y utilizar sus propios argumentos de trabajo, consulte Llamadas a las API de AWS Glue en Python en la guía para desarrolladores.

Para obtener información sobre los argumentos que puede proporcionar a este campo al configurar los trabajos de Spark, consulte el tema Parámetros especiales utilizados por AWS Glue en la guía para desarrolladores.

Para obtener información sobre los argumentos que puede proporcionar a este campo al configurar los trabajos de Ray, consulte Utilizar parámetros de trabajo utilizados en trabajos de Ray en la guía para desarrolladores.

- `NonOverridableArguments`: matriz de mapas de pares clave-valor.

  Cada clave es una cadena UTF-8.

  Cada valor es una cadena UTF-8.

  Los argumentos para este trabajo que no se anulan cuando se proporcionan argumentos de trabajo en una ejecución de trabajos, se especifican como pares nombre-valor.

- `Connections`: un objeto ConnectionsList.

  Las conexiones que se utilizan para este flujo de trabajo.

- `MaxRetries`: número (entero).

  Número máximo de reintentos permitidos para esta tarea si se genera un error.

- `AllocatedCapacity`: número (entero).

  Este parámetro se ha quedado obsoleto. En su lugar, use `MaxCapacity`.

  El número de unidades de procesamiento de AWS Glue datos (DPU) que se van a asignar a este Job. Puede asignar un mínimo de 2 DPU; el valor predeterminado es 10. Una DPU es una medida relativa de la potencia de procesamiento que consta de 4 vCPU de capacidad de cómputo y 16 GB de memoria. Para obtener más información, consulte la página de precios de AWS Glue.

- `Timeout`: número (entero), como mínimo 1.

  El tiempo de espera del flujo de trabajo en minutos. Es el tiempo máximo que una ejecución de trabajo puede consumir recursos antes de que se termine y cambie al estado `TIMEOUT`. El valor predeterminado es 2 880 minutos (48 horas) para los trabajos por lotes.

  Los trabajos de streaming deben tener valores de tiempo de espera inferiores a 7 días o 10 080 minutos. Si el valor se deja en blanco, el trabajo se reiniciará al cabo de 7 días si no se ha

configurado un período de mantenimiento. Si ha configurado un período de mantenimiento, se reiniciará durante el período de mantenimiento a los 7 días.

- `MaxCapacity`: número (doble).

  Para los trabajos de Glue versión 1.0 o anteriores, utilizando el tipo de trabajador estándar, el número de unidades de procesamiento de AWS Glue datos (DPU) que se pueden asignar cuando se ejecuta este trabajo. Una DPU es una medida relativa de la potencia de procesamiento que consta de 4 vCPU de capacidad de cómputo y 16 GB de memoria. Para obtener más información, consulte la página de precios de AWS Glue.

  En los trabajos de Glue versión 2.0+, no puede especificar un `Maximum capacity`. En su lugar, debe especificar un `Worker type` y el `Number of workers`.

  No establezca `MaxCapacity` si utiliza `WorkerType` y `NumberOfWorkers`.

  El valor que se puede asignar a `MaxCapacity` depende de si se está ejecutando un trabajo de shell de Python, un trabajo de ETL de Apache Spark o un trabajo de ETL de streaming de Apache Spark:

  - Cuando especifica un trabajo de shell de Python (`JobCommand.Name`="pythonshell"), puede asignar 0,0625 o 1 DPU. El valor predeterminado es 0,0625 DPU.
  - Cuando especifica un trabajo ETL de Apache Spark (`JobCommand.Name`="glueetl") o un trabajo de ETL de streaming de Apache Spark (`JobCommand.Name`="gluestreaming"), puede asignar de 2 a 100 DPU. El valor predeterminado es 10 DPU. Este tipo de trabajo no puede tener una asignación de DPU fraccionaria.
- `SecurityConfiguration`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la estructura `SecurityConfiguration` que se va a utilizar con este trabajo.
- `Tags`: matriz de mapas de pares de clave-valor, con 50 pares como máximo.

  Cada clave es una cadena UTF-8 con una longitud de entre 1 y 128 bytes.

  Cada valor es una cadena UTF-8 que no tiene más de 256 bytes de largo.

  Etiquetas que se van a utilizar con este trabajo. Puede utilizar etiquetas para limitar el acceso al trabajo. Para obtener más información sobre la inserción de etiquetas AWS Glue, consulte AWS Etiquetar AWS Glue en la guía para desarrolladores.
- `NotificationProperty`: un objeto NotificationProperty.

Especifica las propiedades de configuración de una notificación de flujo de trabajo.

- `GlueVersion`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Custom string pattern #20.

  En los trabajos de Spark, `GlueVersion` determina las versiones de Apache Spark y Python que AWS Glue están disponibles en un trabajo. La versión de Python indica la versión admitida para trabajos de tipo Spark.

  Los trabajos de Ray se deben configurar `GlueVersion` en `4.0` o superior. Sin embargo, las versiones de Ray, Python y bibliotecas adicionales que están disponibles en el trabajo de Ray están determinadas por el parámetro `Runtime` del comando del trabajo.

  Para obtener más información sobre las AWS Glue versiones disponibles y las correspondientes versiones de Spark y Python, consulta la versión de Glue en la guía para desarrolladores.

  Los trabajos que se crean sin especificar una versión de Glue se establecen de forma predeterminada en Glue 0.9.

- `NumberOfWorkers`: número (entero).

  El número de procesos de trabajo de `workerType` definido que se asignan cuando se ejecuta un trabajo.

- `WorkerType`: cadena UTF-8 (valores válidos: `Standard=""` | `G.1X=""` | `G.2X=""` | `G.025X=""` | `G.4X=""` | `G.8X=""` | `Z.2X=""`).

  El tipo de proceso de trabajo predefinido que se asigna cuando se ejecuta un trabajo. Acepta un valor de G.1X, G.2X, G.4X, G.8X o G.025X para los trabajos de Spark. Acepta el valor Z.2X para los trabajos Ray.

  - Para el tipo de trabajador `G.1X`, cada trabajador se asocia a 1 DPU (4 vCPU, 16 GB de memoria) con un disco de 84 GB (aproximadamente 34 GB libres) y proporciona 1 ejecutor por trabajador. Recomendamos este tipo de trabajador para cargas de trabajo como transformaciones de datos, uniones y consultas, ya que ofrece una forma escalable y rentable de ejecutar la mayoría de los trabajos.
  - Para el tipo de trabajador `G.2X`, cada trabajador se asocia a 2 DPU (8 vCPU, 32 GB de memoria) con un disco de 128 GB (aproximadamente 77 GB libres) y proporciona 1 ejecutor por trabajador. Recomendamos este tipo de trabajador para cargas de trabajo como transformaciones de datos, uniones y consultas, ya que ofrece una forma escalable y rentable de ejecutar la mayoría de los trabajos.

- Para el tipo de trabajador `G.4X`, cada trabajador se asocia a 4 DPU (16 vCPU, 64 GB de memoria) con un disco de 256 GB (aproximadamente 235 GB libres) y proporciona 1 ejecutor por trabajador. Recomendamos este tipo de trabajador para los trabajos cuyas cargas de trabajo contengan las transformaciones, agregaciones, uniones y consultas más exigentes. Este tipo de trabajador solo está disponible para los trabajos de Spark ETL de la AWS Glue versión 3.0 o posteriores en AWS las siguientes regiones: EE.UU. Este (Ohio), EE.UU. Este (Norte de Virginia), EE.UU. Oeste (Oregón), Asia Pacífico (Singapur), Asia Pacífico (Sídney), Asia Pacífico (Tokio), Canadá (Central), Europa (Fráncfort), Europa (Irlanda) y Europa (Estocolmo).
- Para el tipo de trabajador `G.8X`, cada trabajador se asocia a 8 DPU (32 GB vCPU, 128 GB de memoria) con un disco de 512 GB (aproximadamente 487 GB libres) y proporciona 1 ejecutor por trabajador. Recomendamos este tipo de trabajador para los trabajos cuyas cargas de trabajo contengan las transformaciones, agregaciones, uniones y consultas más exigentes. Este tipo de trabajador solo está disponible para los trabajos ETL de Spark de la AWS Glue versión 3.0 o posteriores, en las mismas AWS regiones compatibles con el tipo de `G.4X` trabajador.
- Para el tipo de trabajador `G.025X`, cada trabajador se asigna a 0,25 DPU (2 vCPU, 4 GB de memoria) con un disco de 84 GB (aproximadamente 34 GB libres) y proporciona 1 ejecutor por trabajador. Le recomendamos este tipo de proceso de trabajo para trabajos de streaming de bajo volumen. Este tipo de trabajador solo está disponible para los trabajos de streaming de la AWS Glue versión 3.0.
- Para el tipo de trabajador `Z.2X`, cada trabajador se asigna a 2 M-DPU (8 vCPU, 64 GB de memoria) con un disco de 128 GB (aproximadamente 120 GB libres) y proporciona hasta 8 trabajadores de Ray en función del escalador automático.

- `CodeGenConfigurationNodes`: matriz de mapas de pares clave-valor.

  Cada clave es una cadena UTF-8 que coincide con el Custom string pattern #39.

  Cada valor es un objeto A CodeGenConfigurationNode.

  La representación de un gráfico acíclico dirigido en el que se basa tanto el componente visual de Glue Studio como la generación de código de Glue Studio.

- `ExecutionClass`: cadena UTF-8 de 16 bytes de largo como máximo (valores válidos: `FLEX="" | STANDARD=""`).

  Indica si el trabajo se ejecuta con una clase de ejecución estándar o flexible. La clase de ejecución estándar es ideal para cargas de trabajo urgentes que requieren un inicio rápido de los trabajos y recursos dedicados.

La clase de ejecución flexible es adecuada para trabajos no urgentes cuyos momentos de inicio y finalización pueden variar.

Solo se permitirá configurar `ExecutionClass` los trabajos con la AWS Glue versión 3.0 o superior y el tipo `glueetl` de comando`FLEX`. La clase de ejecución flexible está disponible para los trabajos de Spark.

- `SourceControlDetails`: un objeto SourceControlDetails.

  Los detalles de una configuración de control de código de origen para un trabajo, que permite la sincronización de los artefactos del trabajo hacia o desde un repositorio remoto.
- `MaintenanceWindow`: cadena UTF-8 que coincide con el Custom string pattern #30.

  Este campo especifica un día de la semana y una hora para el período de mantenimiento de los trabajos de streaming. AWS Glue realiza actividades de mantenimiento periódicamente. Durante estos períodos de mantenimiento, AWS Glue tendrás que reiniciar tus trabajos de streaming.

  AWS Glue reiniciará el trabajo en un plazo de 3 horas a partir del período de mantenimiento especificado. Por ejemplo, si configura el período de mantenimiento para el lunes a las 10:00 h GMT, sus trabajos se reiniciarán entre las 10:00 h GMT y las 13:00 h GMT.
- `ProfileName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de un perfil de AWS Glue uso asociado al trabajo.

Respuesta

- `Name`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre único que se proporcionó para esta definición de flujo de trabajo.

Errores

- `InvalidInputException`
- `IdempotentParameterMismatchException`
- `AlreadyExistsException`
- `InternalServiceException`

- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`
- `ConcurrentModificationException`

UpdateJob acción (Python: update_job)

Actualiza una definición de trabajo existente. Esta información sobrescribe por completo la definición del trabajo anterior.

Solicitud

- `JobName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre de la definición de trabajo que se va a actualizar.
- `JobUpdate`: obligatorio: objeto JobUpdate.

  Especifica los valores con los que se va a actualizar la definición de flujo de trabajo. La configuración no especificada se elimina o se restablece a los valores predeterminados.
- `ProfileName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de un perfil AWS Glue de uso asociado al trabajo.

Respuesta

- `JobName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Devuelve el nombre de la definición de flujo de trabajo actualizada.

Errores

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`

- `ConcurrentModificationException`

GetJob acción (Python: get_job)

Recupera una definición de trabajo existente.

Solicitud

- `JobName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la definición de flujo de trabajo que se va a recuperar.

Respuesta

- `Job`: un objeto Trabajo.

  La definición de flujo de trabajo solicitada.

Errores

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`

GetJobs acción (Python: get_jobs)

Recupera todas las definiciones de flujo de trabajo actuales.

Solicitud

- `NextToken`: cadena UTF-8.

  Token de continuación si se trata de una llamada de continuidad.
- `MaxResults`: número (entero) que no es inferior a 1 ni es superior a 1000.

  Tamaño máximo de la respuesta.

Respuesta

- `Jobs`: matriz de objetos Trabajo.

  Lista de definiciones de flujo de trabajo.
- `NextToken`: cadena UTF-8.

  Token de continuación si todavía no se han devuelto todas las definiciones de flujo de trabajo.

Errores

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`

DeleteJob acción (Python: delete_job)

Elimina una definición de flujo de trabajo especificada. Si no se encuentra la definición de flujo de trabajo, no se genera una excepción.

Solicitud

- `JobName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la definición de flujo de trabajo que se va a eliminar.

Respuesta

- `JobName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre de la definición de flujo de trabajo que se eliminó.

Errores

- `InvalidInputException`

- `InternalServiceException`
- `OperationTimeoutException`

ListJobs acción (Python: list_jobs)

Recupera los nombres de todos los recursos de trabajo de esta AWS cuenta o los recursos con la etiqueta especificada. Esta operación permite ver qué recursos están disponibles en la cuenta y sus nombres.

Esta operación toma el campo `Tags` opcional, que se puede utilizar como filtro en la respuesta para que los recursos etiquetados se devuelvan agrupados. Si decide utilizar el filtrado de etiquetas, solo se devolverán los recursos con la etiqueta especificada.

Solicitud

- `NextToken`: cadena UTF-8.

  Token de continuación, si se trata de una solicitud de continuidad.
- `MaxResults`: número (entero) que no es inferior a 1 ni es superior a 1000.

  Tamaño máximo de una lista que se devolverá.
- `Tags`: matriz de mapas de pares de clave-valor, con 50 pares como máximo.

  Cada clave es una cadena UTF-8 con una longitud de entre 1 y 128 bytes.

  Cada valor es una cadena UTF-8 que no tiene más de 256 bytes de largo.

  Especifica que se devuelvan solamente los recursos etiquetados.

Respuesta

- `JobNames`: matriz de cadenas UTF-8.

  Nombres de todos los trabajos de la cuenta o de los trabajos con las etiquetas especificadas.
- `NextToken`: cadena UTF-8.

  Token de continuación, si la lista devuelta no contiene la última métrica disponible.

Errores

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`

BatchGetJobs acción (Python: batch_get_jobs)

Devuelve la lista de metadatos de recursos de una determinada lista de nombres de trabajos. Después de llamar a la operación `ListJobs`, puede llamar a esta operación para obtener acceso a los datos a los que ha concedido permisos. Esta operación admite todos los permisos de IAM, incluidas las condiciones de permisos que utilizan etiquetas.

Solicitud

- `JobNames` – Obligatorio: una matriz de cadenas UTF-8.

  Una lista de nombres de trabajos, que pueden ser los nombres devueltos en la operación `ListJobs`.

Respuesta

- `Jobs`: matriz de objetos Trabajo.

  Lista de definiciones de flujo de trabajo.
- `JobsNotFound`: matriz de cadenas UTF-8.

  No se encuentra ninguna lista de nombres de trabajos.

Errores

- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`

Ejecuciones de trabajo

La API Jobs Runs describe los tipos de datos y la API relacionados con el inicio, la detención o la visualización de las ejecuciones de trabajos y el restablecimiento de los marcadores de trabajos, en. AWS Glue El historial de ejecución de trabajos está disponible durante 90 días para su flujo de trabajo y ejecución de trabajos.

Tipos de datos

- JobRun estructura
- Estructura Predecessor
- JobBookmarkEntry estructura
- BatchStopJobRunSuccessfulSubmission estructura
- BatchStopJobRunError estructura
- NotificationProperty estructura

JobRun estructura

Contiene información acerca de una ejecución de flujo de trabajo.

Campos

- `Id`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID de la ejecución de este flujo de trabajo.
- `Attempt`: número (entero).

  El número del intento de ejecución de este flujo de trabajo.
- `PreviousRunId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID de la ejecución anterior de este trabajo. Por ejemplo, el`JobRunId` especificado en la acción `StartJobRun`.
- `TriggerName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre del disparador que inició esta ejecución de flujo de trabajo.

- `JobName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la definición de flujo de trabajo que se utiliza en esta ejecución.

- `JobMode`: cadena UTF-8 (valores válidos: `SCRIPT=""` | `VISUAL=""` | `NOTEBOOK=""`).

  Un modo que describe cómo se creó un trabajo. Los valores válidos son:

  - `SCRIPT`- El trabajo se creó con el editor de scripts de AWS Glue Studio.
  - `VISUAL`- El trabajo se creó con el editor visual de AWS Glue Studio.
  - `NOTEBOOK`: El trabajo se creó con un cuaderno de sesiones interactivo.

  Cuando el campo `JobMode` no aparece o es nulo, se asigna `SCRIPT` como valor predeterminado.

- `StartedOn`: marca temporal.

  La fecha y la hora en que se inició la ejecución de este flujo de trabajo.

- `LastModifiedOn`: marca temporal.

  Última modificación de la ejecución de este trabajo.

- `CompletedOn`: marca temporal.

  La fecha y la hora en que se completó la ejecución de este trabajo.

- `JobRunState`: Cadena UTF-8 (valores válidos: `STARTING` | `RUNNING` | `STOPPING` | `STOPPED` | `SUCCEEDED` | `FAILED` | `TIMEOUT` | `ERROR` | `WAITING` | `EXPIRED`).

  El estado actual de la ejecución de flujo de trabajo. Para obtener más información sobre los estados de los trabajos que han terminado de forma anormal, consulte Estados de ejecución de trabajos de AWS Glue.

- `Arguments`: matriz de mapas de pares clave-valor.

  Cada clave es una cadena UTF-8.

  Cada valor es una cadena UTF-8.

  Los argumentos de flujo de trabajo asociados a esta ejecución. En esta ejecución del trabajo, sustituyen a los argumentos predeterminados definidos en la propia definición del trabajo.

  Aquí puede especificar los argumentos que consume su propio script de ejecución de tareas, así como los argumentos que consume el AWS Glue mismo.

Es posible que se registren los argumentos del trabajo. No utilice secretos con formato de texto no cifrado como argumentos. Recupera los secretos de una AWS Glue Conexión AWS Secrets Manager u otro mecanismo de gestión de secretos si pretendes mantenerlos en el Job.

Para obtener información acerca de cómo especificar y utilizar sus propios argumentos de trabajo, consulte Llamadas a las API de AWS Glue en Python en la guía para desarrolladores.

Para obtener información sobre los argumentos que puede proporcionar a este campo al configurar los trabajos de Spark, consulte el tema Parámetros especiales utilizados por AWS Glue en la guía para desarrolladores.

Para obtener información sobre los argumentos que puede proporcionar a este campo al configurar los trabajos de Ray, consulte Utilizar parámetros de trabajo utilizados en trabajos de Ray en la guía para desarrolladores.

- `ErrorMessage`: cadena UTF-8.

  Un mensaje de error asociado a la ejecución de este flujo de trabajo.

- `PredecessorRuns`: matriz de objetos Predecessor.

  Una lista de predecesores a la ejecución de este flujo de trabajo.

- `AllocatedCapacity`: número (entero).

  Este campo está obsoleto. En su lugar, use `MaxCapacity`.

  El número de unidades de procesamiento de AWS Glue datos (DPU) asignadas a esto JobRun. Se pueden asignar entre 2 y 100 DPU; el valor predeterminado es 10. Una DPU es una medida relativa de la potencia de procesamiento que consta de 4 vCPU de capacidad de cómputo y 16 GB de memoria. Para obtener más información, consulte la página de precios de AWS Glue.

- `ExecutionTime`: número (entero).

  El periodo de tiempo (en segundos) que la ejecución de flujo de trabajo consumió recursos.

- `Timeout`: número (entero), como mínimo 1.

  Tiempo de espera de `JobRun` en minutos. Es el tiempo máximo que una ejecución de trabajo puede consumir recursos antes de que se termine y cambie al estado `TIMEOUT`. Este valor sustituye el valor de tiempo de espera establecido en el flujo de trabajo principal.

Los trabajos de streaming deben tener valores de tiempo de espera inferiores a 7 días o 10 080 minutos. Si el valor se deja en blanco, el trabajo se reiniciará al cabo de 7 días si no se ha configurado un período de mantenimiento. Si ha configurado un período de mantenimiento, se reiniciará durante el período de mantenimiento a los 7 días.

- `MaxCapacity`: número (doble).

  Para los trabajos de Glue versión 1.0 o anteriores, utilizando el tipo de trabajador estándar, el número de unidades de procesamiento de AWS Glue datos (DPU) que se pueden asignar cuando se ejecuta este trabajo. Una DPU es una medida relativa de la potencia de procesamiento que consta de 4 vCPU de capacidad de cómputo y 16 GB de memoria. Para obtener más información, consulte la página de precios de AWS Glue.

  En los trabajos de Glue versión 2.0+, no puede especificar un `Maximum capacity`. En su lugar, debe especificar un `Worker type` y el `Number of workers`.

  No establezca `MaxCapacity` si utiliza `WorkerType` y `NumberOfWorkers`.

  El valor que se puede asignar a `MaxCapacity` depende de si se está ejecutando un trabajo de shell de Python, un trabajo de ETL de Apache Spark o un trabajo de ETL de streaming de Apache Spark:

  - Cuando especifica un trabajo de shell de Python (`JobCommand.Name`="pythonshell"), puede asignar 0,0625 o 1 DPU. El valor predeterminado es 0,0625 DPU.
  - Cuando especifica un trabajo ETL de Apache Spark (`JobCommand.Name`="glueetl") o un trabajo de ETL de streaming de Apache Spark (`JobCommand.Name`="gluestreaming"), puede asignar de 2 a 100 DPU. El valor predeterminado es 10 DPU. Este tipo de trabajo no puede tener una asignación de DPU fraccionaria.
- `WorkerType`: cadena UTF-8 (valores válidos: `Standard=""` | `G.1X=""` | `G.2X=""` | `G.025X=""` | `G.4X=""` | `G.8X=""` | `Z.2X=""`).

  El tipo de proceso de trabajo predefinido que se asigna cuando se ejecuta un trabajo. Acepta un valor de G.1X, G.2X, G.4X, G.8X o G.025X para los trabajos de Spark. Acepta el valor Z.2X para los trabajos Ray.

  - Para el tipo de trabajador `G.1X`, cada trabajador se asocia a 1 DPU (4 vCPU, 16 GB de memoria) con un disco de 84 GB (aproximadamente 34 GB libres) y proporciona 1 ejecutor por trabajador. Recomendamos este tipo de trabajador para cargas de trabajo como

transformaciones de datos, uniones y consultas, ya que ofrece una forma escalable y rentable de ejecutar la mayoría de los trabajos.

- Para el tipo de trabajador `G.2X`, cada trabajador se asocia a 2 DPU (8 vCPU, 32 GB de memoria) con un disco de 128 GB (aproximadamente 77 GB libres) y proporciona 1 ejecutor por trabajador. Recomendamos este tipo de trabajador para cargas de trabajo como transformaciones de datos, uniones y consultas, ya que ofrece una forma escalable y rentable de ejecutar la mayoría de los trabajos.
- Para el tipo de trabajador `G.4X`, cada trabajador se asocia a 4 DPU (16 vCPU, 64 GB de memoria) con un disco de 256 GB (aproximadamente 235 GB libres) y proporciona 1 ejecutor por trabajador. Recomendamos este tipo de trabajador para los trabajos cuyas cargas de trabajo contengan las transformaciones, agregaciones, uniones y consultas más exigentes. Este tipo de trabajador solo está disponible para los trabajos de Spark ETL de la AWS Glue versión 3.0 o posteriores en AWS las siguientes regiones: EE.UU. Este (Ohio), EE.UU. Este (Norte de Virginia), EE.UU. Oeste (Oregón), Asia Pacífico (Singapur), Asia Pacífico (Sídney), Asia Pacífico (Tokio), Canadá (Central), Europa (Fráncfort), Europa (Irlanda) y Europa (Estocolmo).
- Para el tipo de trabajador `G.8X`, cada trabajador se asocia a 8 DPU (32 GB vCPU, 128 GB de memoria) con un disco de 512 GB (aproximadamente 487 GB libres) y proporciona 1 ejecutor por trabajador. Recomendamos este tipo de trabajador para los trabajos cuyas cargas de trabajo contengan las transformaciones, agregaciones, uniones y consultas más exigentes. Este tipo de trabajador solo está disponible para los trabajos ETL de Spark de la AWS Glue versión 3.0 o posteriores, en las mismas AWS regiones compatibles con el tipo de `G.4X` trabajador.
- Para el tipo de trabajador `G.025X`, cada trabajador se asigna a 0,25 DPU (2 vCPU, 4 GB de memoria) con un disco de 84 GB (aproximadamente 34 GB libres) y proporciona 1 ejecutor por trabajador. Le recomendamos este tipo de proceso de trabajo para trabajos de streaming de bajo volumen. Este tipo de trabajador solo está disponible para los trabajos de streaming de la AWS Glue versión 3.0.
- Para el tipo de trabajador `Z.2X`, cada trabajador se asigna a 2 M-DPU (8 vCPU, 64 GB de memoria) con un disco de 128 GB (aproximadamente 120 GB libres) y proporciona hasta 8 trabajadores de Ray en función del escalador automático.

- `NumberOfWorkers`: número (entero).

  El número de procesos de trabajo de `workerType` definido que se asignan cuando se ejecuta un trabajo.

- `SecurityConfiguration`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

El nombre de la estructura `SecurityConfiguration` que se va a utilizar con este trabajo.

- `LogGroupName`: cadena UTF-8.

  El nombre del grupo de registros para un registro seguro que se puede cifrar en el lado del servidor en Amazon CloudWatch mediante. AWS KMS Este nombre puede ser `/aws-glue/jobs/`, en cuyo caso el cifrado predeterminado es `NONE`. Si añade un nombre de rol y el nombre `SecurityConfiguration` (en otras palabras, `/aws-glue/jobs-yourRoleName-yourSecurityConfigurationName/`), entonces dicha configuración de seguridad se utiliza para cifrar el grupo de registros.

- `NotificationProperty`: un objeto NotificationProperty.

  Especifica las propiedades de configuración de una notificación de ejecución de trabajo.

- `GlueVersion`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Custom string pattern #20.

  En los trabajos de Spark, `GlueVersion` determina las versiones de Apache Spark y Python que AWS Glue están disponibles en un trabajo. La versión de Python indica la versión admitida para trabajos de tipo Spark.

  Los trabajos de Ray se deben configurar `GlueVersion` en `4.0` o superior. Sin embargo, las versiones de Ray, Python y bibliotecas adicionales que están disponibles en el trabajo de Ray están determinadas por el parámetro `Runtime` del comando del trabajo.

  Para obtener más información sobre las AWS Glue versiones disponibles y las correspondientes versiones de Spark y Python, consulta la versión de Glue en la guía para desarrolladores.

  Los trabajos que se crean sin especificar una versión de Glue se establecen de forma predeterminada en Glue 0.9.

- `DPUSeconds`: número (doble).

  Este campo puede configurarse para ejecuciones de trabajos con clase de ejecución `FLEX` o cuando el escalado automático está activado, y representa el tiempo total que estuvo activo cada ejecutor durante el ciclo de vida de una ejecución de trabajo en segundos, multiplicado por un factor de DPU (1 para trabajadores `G.1X`, 2 para `G.2X`, o 0,25 para `G.025X`). Este valor puede ser diferente del de `executionEngineRuntime` * `MaxCapacity`, como en el caso de los trabajos de Auto Scaling, ya que el número de ejecutores que están activos en un momento determinado

puede ser inferior a `MaxCapacity`. Por lo tanto, es posible que el valor de `DPUSeconds` sea menor que `executionEngineRuntime` * `MaxCapacity`.

- `ExecutionClass`: cadena UTF-8 de 16 bytes de largo como máximo (valores válidos: `FLEX=""` | `STANDARD=""`).

  Indica si el trabajo se ejecuta con una clase de ejecución estándar o flexible. La clase de ejecución estándar es ideal para cargas de trabajo urgentes que requieren un inicio rápido de los trabajos y recursos dedicados.

  La clase de ejecución flexible es adecuada para trabajos no urgentes cuyos momentos de inicio y finalización pueden variar.

  Solo se `glueetl` podrán configurar `ExecutionClass` los trabajos con la AWS Glue versión 3.0 o superior y el tipo de comando`FLEX`. La clase de ejecución flexible está disponible para los trabajos de Spark.

- `MaintenanceWindow`: cadena UTF-8 que coincide con el Custom string pattern #30.

  Este campo especifica un día de la semana y una hora para el período de mantenimiento de los trabajos de streaming. AWS Glue realiza actividades de mantenimiento periódicamente. Durante estos períodos de mantenimiento, AWS Glue tendrás que reiniciar tus trabajos de streaming.

  AWS Glue reiniciará el trabajo en un plazo de 3 horas a partir del período de mantenimiento especificado. Por ejemplo, si configura el período de mantenimiento para el lunes a las 10:00 h GMT, sus trabajos se reiniciarán entre las 10:00 h GMT y las 13:00 h GMT.

- `ProfileName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de un perfil de AWS Glue uso asociado a la ejecución de la tarea.

Estructura Predecessor

Una ejecución de flujo de trabajo que se usó en el predicado de un disparador condicional que activó la ejecución de este flujo de trabajo.

Campos

- `JobName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

El nombre de la definición de flujo de trabajo que usa la ejecución de flujo de trabajo del predecesor.

- `RunId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID de ejecución del flujo de trabajo de la ejecución de flujo de trabajo del predecesor.

JobBookmarkEntry estructura

Define un momento en el que un trabajo puede reanudar el procesamiento.

Campos

- `JobName`: cadena UTF-8.

  Nombre del trabajo en cuestión.
- `Version`: número (entero).

  La versión del trabajo.
- `Run`: número (entero).

  El número de ID de ejecución.
- `Attempt`: número (entero).

  El número de ID de intento.
- `PreviousRunId`: cadena UTF-8.

  El identificador de ejecución único asociado a esta ejecución.
- `RunId`: cadena UTF-8.

  El número de ID de ejecución.
- `JobBookmark`: cadena UTF-8.

  El propio marcador.

BatchStopJobRunSuccessfulSubmission estructura

Registra una solicitud correcta para detener un objeto `JobRun` especificado.

Campos

- `JobName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la definición de flujo de trabajo usada en la ejecución de flujo de trabajo que se detuvo.
- `JobRunId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El `JobRunId` de la ejecución de trabajo que se detuvo.

BatchStopJobRunError estructura

Registra un error que se produjo al intentar detener una ejecución de flujo de trabajo especificada.

Campos

- `JobName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la definición de trabajo usada en la ejecución de trabajo en cuestión.
- `JobRunId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El `JobRunId` de la ejecución de trabajo en cuestión.
- `ErrorDetail`: un objeto ErrorDetail.

  Especifica los detalles acerca del error que se encontró.

NotificationProperty estructura

Especifica las propiedades de configuración de una notificación.

Campos

- `NotifyDelayAfter`: número (entero), como mínimo 1.

  Después de que comience una ejecución de flujo de trabajo, el número de minutos que se debe esperar antes de enviar una notificación de retraso de ejecución de un flujo de trabajo.

Operaciones

- StartJobRun acción (Python: start_job_run)
- BatchStopJobRun acción (Python: batch_stop_job_run)
- GetJobRun acción (Python: get_job_run)
- GetJobRuns acción (Python: get_job_runs)
- GetJobBookmark acción (Python: get_job_bookmark)
- GetJobBookmarks acción (Python: get_job_bookmarks)
- ResetJobBookmark acción (Python: reset_job_bookmark)

StartJobRun acción (Python: start_job_run)

Inicia una ejecución de flujo de trabajo con una definición de flujo de trabajo.

Solicitud

- `JobName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la definición de flujo de trabajo que se va a usar.
- `JobRunId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID de una `JobRun` anterior para volver a intentarlo.
- `Arguments`: matriz de mapas de pares clave-valor.

  Cada clave es una cadena UTF-8.

  Cada valor es una cadena UTF-8.

  Los argumentos de flujo de trabajo asociados a esta ejecución. En esta ejecución del trabajo, sustituyen a los argumentos predeterminados definidos en la propia definición del trabajo.

  Aquí puede especificar los argumentos que consume su propio script de ejecución de tareas, así como los argumentos que consume él mismo. AWS Glue

Es posible que se registren los argumentos del trabajo. No utilice secretos con formato de texto no cifrado como argumentos. Recupera los secretos de una AWS Glue Conexión AWS Secrets Manager u otro mecanismo de gestión de secretos si pretendes mantenerlos en el Job.

Para obtener información acerca de cómo especificar y utilizar sus propios argumentos de trabajo, consulte Llamadas a las API de AWS Glue en Python en la guía para desarrolladores.

Para obtener información sobre los argumentos que puede proporcionar a este campo al configurar los trabajos de Spark, consulte el tema Parámetros especiales utilizados por AWS Glue en la guía para desarrolladores.

Para obtener información sobre los argumentos que puede proporcionar a este campo al configurar los trabajos de Ray, consulte Utilizar parámetros de trabajo utilizados en trabajos de Ray en la guía para desarrolladores.

- `AllocatedCapacity`: número (entero).

  Este campo está obsoleto. En su lugar, use `MaxCapacity`.

  El número de unidades de procesamiento de AWS Glue datos (DPU) que se van a asignar a esto JobRun. Puede asignar un mínimo de 2 DPU; el valor predeterminado es 10. Una DPU es una medida relativa de la potencia de procesamiento que consta de 4 vCPU de capacidad de cómputo y 16 GB de memoria. Para obtener más información, consulte la página de precios de AWS Glue.

- `Timeout`: número (entero), como mínimo 1.

  Tiempo de espera de `JobRun` en minutos. Es el tiempo máximo que una ejecución de trabajo puede consumir recursos antes de que se termine y cambie al estado `TIMEOUT`. Este valor sustituye el valor de tiempo de espera establecido en el flujo de trabajo principal.

  Los trabajos de streaming deben tener valores de tiempo de espera inferiores a 7 días o 10 080 minutos. Si el valor se deja en blanco, el trabajo se reiniciará al cabo de 7 días si no se ha configurado un período de mantenimiento. Si ha configurado un período de mantenimiento, se reiniciará durante el período de mantenimiento a los 7 días.

- `MaxCapacity`: número (doble).

  Para los trabajos de Glue versión 1.0 o anteriores, utilizando el tipo de trabajador estándar, el número de unidades de procesamiento de AWS Glue datos (DPU) que se pueden asignar cuando se ejecuta este trabajo. Una DPU es una medida relativa de la potencia de procesamiento que

consta de 4 vCPU de capacidad de cómputo y 16 GB de memoria. Para obtener más información, consulte la página de precios de AWS Glue.

En los trabajos de Glue versión 2.0+, no puede especificar un `Maximum capacity`. En su lugar, debe especificar un `Worker type` y el `Number of workers`.

No establezca `MaxCapacity` si utiliza `WorkerType` y `NumberOfWorkers`.

El valor que se puede asignar a `MaxCapacity` depende de si se está ejecutando un trabajo de shell de Python, un trabajo de ETL de Apache Spark o un trabajo de ETL de streaming de Apache Spark:

- Cuando especifica un trabajo de shell de Python (`JobCommand.Name`="pythonshell"), puede asignar 0,0625 o 1 DPU. El valor predeterminado es 0,0625 DPU.
- Cuando especifica un trabajo ETL de Apache Spark (`JobCommand.Name`="glueetl") o un trabajo de ETL de streaming de Apache Spark (`JobCommand.Name`="gluestreaming"), puede asignar de 2 a 100 DPU. El valor predeterminado es 10 DPU. Este tipo de trabajo no puede tener una asignación de DPU fraccionaria.

- `SecurityConfiguration`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la estructura `SecurityConfiguration` que se va a utilizar con este trabajo.

- `NotificationProperty`: un objeto NotificationProperty.

  Especifica las propiedades de configuración de una notificación de ejecución de trabajo.

- `WorkerType`: cadena UTF-8 (valores válidos: `Standard=""` | `G.1X=""` | `G.2X=""` | `G.025X=""` | `G.4X=""` | `G.8X=""` | `Z.2X=""`).

  El tipo de proceso de trabajo predefinido que se asigna cuando se ejecuta un trabajo. Acepta un valor de G.1X, G.2X, G.4X, G.8X o G.025X para los trabajos de Spark. Acepta el valor Z.2X para los trabajos Ray.

  - Para el tipo de trabajador `G.1X`, cada trabajador se asocia a 1 DPU (4 vCPU, 16 GB de memoria) con un disco de 84 GB (aproximadamente 34 GB libres) y proporciona 1 ejecutor por trabajador. Recomendamos este tipo de trabajador para cargas de trabajo como transformaciones de datos, uniones y consultas, ya que ofrece una forma escalable y rentable de ejecutar la mayoría de los trabajos.
  - Para el tipo de trabajador `G.2X`, cada trabajador se asocia a 2 DPU (8 vCPU, 32 GB de memoria) con un disco de 128 GB (aproximadamente 77 GB libres) y proporciona 1

ejecutor por trabajador. Recomendamos este tipo de trabajador para cargas de trabajo como transformaciones de datos, uniones y consultas, ya que ofrece una forma escalable y rentable de ejecutar la mayoría de los trabajos.

- Para el tipo de trabajador `G.4X`, cada trabajador se asocia a 4 DPU (16 vCPU, 64 GB de memoria) con un disco de 256 GB (aproximadamente 235 GB libres) y proporciona 1 ejecutor por trabajador. Recomendamos este tipo de trabajador para los trabajos cuyas cargas de trabajo contengan las transformaciones, agregaciones, uniones y consultas más exigentes. Este tipo de trabajador solo está disponible para los trabajos de Spark ETL de la AWS Glue versión 3.0 o posteriores en AWS las siguientes regiones: EE.UU. Este (Ohio), EE.UU. Este (Norte de Virginia), EE.UU. Oeste (Oregón), Asia Pacífico (Singapur), Asia Pacífico (Sídney), Asia Pacífico (Tokio), Canadá (Central), Europa (Fráncfort), Europa (Irlanda) y Europa (Estocolmo).
- Para el tipo de trabajador `G.8X`, cada trabajador se asocia a 8 DPU (32 GB vCPU, 128 GB de memoria) con un disco de 512 GB (aproximadamente 487 GB libres) y proporciona 1 ejecutor por trabajador. Recomendamos este tipo de trabajador para los trabajos cuyas cargas de trabajo contengan las transformaciones, agregaciones, uniones y consultas más exigentes. Este tipo de trabajador solo está disponible para los trabajos ETL de Spark de la AWS Glue versión 3.0 o posteriores, en las mismas AWS regiones compatibles con el tipo de `G.4X` trabajador.
- Para el tipo de trabajador `G.025X`, cada trabajador se asigna a 0,25 DPU (2 vCPU, 4 GB de memoria) con un disco de 84 GB (aproximadamente 34 GB libres) y proporciona 1 ejecutor por trabajador. Le recomendamos este tipo de proceso de trabajo para trabajos de streaming de bajo volumen. Este tipo de trabajador solo está disponible para los trabajos de streaming de la AWS Glue versión 3.0.
- Para el tipo de trabajador `Z.2X`, cada trabajador se asigna a 2 M-DPU (8 vCPU, 64 GB de memoria) con un disco de 128 GB (aproximadamente 120 GB libres) y proporciona hasta 8 trabajadores de Ray en función del escalador automático.

- `NumberOfWorkers`: número (entero).

  El número de procesos de trabajo de `workerType` definido que se asignan cuando se ejecuta un trabajo.

- `ExecutionClass`: cadena UTF-8 de 16 bytes de largo como máximo (valores válidos: `FLEX=""` | `STANDARD=""`).

  Indica si el trabajo se ejecuta con una clase de ejecución estándar o flexible. La clase de ejecución estándar es ideal para cargas de trabajo urgentes que requieren un inicio rápido de los trabajos y recursos dedicados.

La clase de ejecución flexible es adecuada para trabajos no urgentes cuyos momentos de inicio y finalización pueden variar.

Solo se permitirá configurar `ExecutionClass` los trabajos con la AWS Glue versión 3.0 o superior y el tipo `glueetl` de comando`FLEX`. La clase de ejecución flexible está disponible para los trabajos de Spark.

- `ProfileName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de un perfil de AWS Glue uso asociado a la ejecución del trabajo.

Respuesta

- `JobRunId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID asignado a la ejecución de este flujo de trabajo.

Errores

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`
- `ConcurrentRunsExceededException`

BatchStopJobRun acción (Python: batch_stop_job_run)

Detiene una o varias ejecuciones de flujo de trabajo para una definición flujo de trabajo especificada.

Solicitud

- `JobName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

El nombre de la definición de flujo de trabajo para la que se detienen las ejecuciones de flujo de trabajo.

- `JobRunIds` – Obligatorio: matriz de cadenas UTF-8, con una cadena como mínimo y 25 cadenas como máximo.

  Una lista de los `JobRunIds` que se deben detener para esa definición de trabajo.

Respuesta

- `SuccessfulSubmissions`: matriz de objetos BatchStopJobRunSuccessfulSubmission.

  Una lista de las que se enviaron correctamente para detenerlas JobRuns .

- `Errors`: matriz de objetos BatchStopJobRunError.

  Una lista de los errores que se encontraron al intentar detener objetos `JobRuns`, incluidos el `JobRunId` para el que se encontró cada error y los detalles acerca del error.

Errores

- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

GetJobRun acción (Python: get_job_run)

Recupera los metadatos para una ejecución de flujo de trabajo especificada. El historial de ejecución de trabajos está disponible durante 90 días para su flujo de trabajo y ejecución de trabajos.

Solicitud

- `JobName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre de la definición de flujo de trabajo que se ejecuta.

- `RunId`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

El ID de la ejecución de flujo de trabajo.

- `PredecessorsIncluded`: booleano.

  True si una lista de ejecuciones del predecesor debe devolverse.

Respuesta

- `JobRun`: un objeto JobRun.

  Los metadatos de ejecución de flujo de trabajo solicitados.

Errores

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`

GetJobRuns acción (Python: get_job_runs)

Recupera los metadatos para todas las ejecuciones de una definición de flujo de trabajo especificada.

Solicitud

- `JobName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la definición de flujo de trabajo para la que se recuperarán todas las ejecuciones de flujo de trabajo.

- `NextToken`: cadena UTF-8.

  Token de continuación si se trata de una llamada de continuidad.

- `MaxResults`: número (entero), mayor que 1 y menor que 200.

  Tamaño máximo de la respuesta.

Respuesta

- `JobRuns`: matriz de objetos JobRun.

  Una lista de objetos de metadatos de ejecución de trabajo

- `NextToken`: cadena UTF-8.

  Token de continuación, si no se han devuelto todas las ejecuciones de trabajos solicitadas.

Errores

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`

GetJobBookmark acción (Python: get_job_bookmark)

Devuelve información sobre una entrada de marcador de trabajo.

Para más información acerca de la habilitación y el uso de marcadores de trabajo, consulte:

- Seguimiento de los datos procesados mediante marcadores de trabajo
- Parámetros de trabajo utilizados por AWS Glue
- Estructura de trabajo

Solicitud

- `JobName` – Obligatorio: cadena UTF-8.

  Nombre del trabajo en cuestión.

- `Version`: número (entero).

  La versión del trabajo.

- `RunId`: cadena UTF-8.

  El identificador de ejecución único asociado a esta ejecución.

Respuesta

- `JobBookmarkEntry`: un objeto JobBookmarkEntry.

  Estructura que define un punto en el que un trabajo puede reanudar el procesamiento.

Errores

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `ValidationException`

GetJobBookmarks acción (Python: get_job_bookmarks)

Devuelve información sobre las entradas de marcador de trabajo. La lista se ordena en números de versión decrecientes.

Para más información acerca de la habilitación y el uso de marcadores de trabajo, consulte:

- Seguimiento de los datos procesados mediante marcadores de trabajo
- Parámetros de trabajo utilizados por AWS Glue
- Estructura de trabajo

Solicitud

- `JobName` – Obligatorio: cadena UTF-8.

  Nombre del trabajo en cuestión.
- `MaxResults`: número (entero).

  Tamaño máximo de la respuesta.
- `NextToken`: número (entero).

  Token de continuación si se trata de una llamada de continuidad.

Respuesta

- `JobBookmarkEntries`: matriz de objetos JobBookmarkEntry.

  Una lista de entradas de marcador de trabajo que define un punto en el que un trabajo puede reanudar su procesamiento.

- `NextToken`: número (entero).

  Un token de continuación, que tiene un valor de 1 si se devuelven todas las entradas, o mayor que 1 si no se han devuelto todas las ejecuciones de trabajo solicitadas.

Errores

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`

ResetJobBookmark acción (Python: reset_job_bookmark)

Restablece una entrada de marcador.

Para más información acerca de la habilitación y el uso de marcadores de trabajo, consulte:

- Seguimiento de los datos procesados mediante marcadores de trabajo
- Parámetros de trabajo utilizados por AWS Glue
- Estructura de trabajo

Solicitud

- `JobName` – Obligatorio: cadena UTF-8.

  Nombre del trabajo en cuestión.

- `RunId`: cadena UTF-8.

  El identificador de ejecución único asociado a esta ejecución.

Respuesta

- `JobBookmarkEntry`: un objeto JobBookmarkEntry.

  La entrada de marcador de restablecimiento.

Errores

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

Desencadenadores

La API de desencadenadores describe los tipos de datos y la API relacionados con la creación, actualización o eliminación, y el inicio y la detención de desencadenadores de trabajos en AWS Glue.

Tipos de datos

- Estructura del desencadenador
- Estructura de TriggerUpdate
- Estructura de Predicate
- Estructura de Condition
- Estructura de acción
- Estructura EventBatchingCondition

Estructura del desencadenador

Información sobre un disparador específico.

Campos

- `Name`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre del desencadenador.

- `WorkflowName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre del flujo de trabajo asociado con el desencadenador.

- `Id`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Reservado para uso futuro.

- `Type`: cadena UTF-8 (valores válidos: `SCHEDULED` | `CONDITIONAL` | `ON_DEMAND` | `EVENT`).

  Tipo de desencadenador en cuestión.

- `State`: cadena UTF-8 (valores válidos: `CREATING` | `CREATED` | `ACTIVATING` | `ACTIVATED` | `DEACTIVATING` | `DEACTIVATED` | `DELETING` | `UPDATING`).

  Estado actual del disparador.

- `Description`: cadena de descripción de un máximo de 2048 bytes de largo, que coincide con el URI address multi-line string pattern.

  Descripción de este desencadenador.

- `Schedule`: cadena UTF-8.

  Expresión `cron` utilizada para especificar el programa (consulte Programaciones basadas en tiempo para trabajos y rastreadores). Por ejemplo, para ejecutar algo todos los días a las 12:15 UTC, especifique: `cron(15 12 * * ? *)`.

- `Actions`: matriz de objetos Acción.

  Acciones iniciadas por este desencadenador.

- `Predicate`: un objeto Predicate.

  Predicado de este desencadenador, que define el momento en que se desencadenará.

- `EventBatchingCondition`: un objeto EventBatchingCondition.

  Condición del lote que debe cumplirse (número especificado de eventos recibidos o ventana de tiempo del lote vencida) antes de que se desencadene el evento EventBridge.

Estructura de TriggerUpdate

Una estructura usada para proporcionar información que se emplea para actualizar un disparador. Este objeto actualizará la definición del disparador anterior sobrescribiéndola por completo.

Campos

- `Name`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Reservado para uso futuro.

- `Description`: cadena de descripción de un máximo de 2048 bytes de largo, que coincide con el URI address multi-line string pattern.

  Descripción de este desencadenador.

- `Schedule`: cadena UTF-8.

  Expresión `cron` utilizada para especificar el programa (consulte Programaciones basadas en tiempo para trabajos y rastreadores). Por ejemplo, para ejecutar algo todos los días a las 12:15 UTC, especifique: `cron(15 12 * * ? *)`.

- `Actions`: matriz de objetos Acción.

  Acciones iniciadas por este desencadenador.

- `Predicate`: un objeto Predicate.

  Predicado de este desencadenador, que define el momento en que se desencadenará.

- `EventBatchingCondition`: un objeto EventBatchingCondition.

  Condición del lote que debe cumplirse (número especificado de eventos recibidos o ventana de tiempo del lote vencida) antes de que se desencadene el evento EventBridge.

Estructura de Predicate

Define el predicado del disparador, que determina cuándo se activa este.

Campos

- `Logical`: cadena UTF-8 (valores válidos: `AND` | `ANY`).

Un campo opcional si solo se indica una condición. Si se indican varias condiciones, este campo es obligatorio.

- `Conditions`: matriz de objetos Condición.

  Lista de condiciones que determinan cuándo se desencadenará el disparador.

Estructura de Condition

Define una condición en la que se desencadena un desencadenador.

Campos

- `LogicalOperator`: cadena UTF-8 (valores válidos: `EQUALS`).

  Operador lógico.

- `JobName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre del trabajo a cuyo `JobRuns` se aplica esta condición y en el que espera este disparador.

- `State`: Cadena UTF-8 (valores válidos: `STARTING` | `RUNNING` | `STOPPING` | `STOPPED` | `SUCCEEDED` | `FAILED` | `TIMEOUT` | `ERROR` | `WAITING` | `EXPIRED`).

  El estado de la condición. Actualmente, los únicos estados de trabajo que un desencadenador puede escuchar son `SUCCEEDED`, `STOPPED`, `FAILED`, y `TIMEOUT`. Los únicos estados de rastreadores que un disparador puede escuchar son `SUCCEEDED`, `FAILED`, y `CANCELLED`.

- `CrawlerName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre del rastreador al que se aplica esta condición.

- `CrawlState`: cadena UTF-8 (valores válidos: `RUNNING` | `CANCELLING` | `CANCELLED` | `SUCCEEDED` | `FAILED` | `ERROR`).

  El estado del rastreador al que se aplica esta condición.

Estructura de acción

Define la acción que un desencadenador iniciará.

Campos

- `JobName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre del trabajo que se ejecutará.
- `Arguments`: matriz de mapas de pares clave-valor.

  Cada clave es una cadena UTF-8.

  Cada valor es una cadena UTF-8.

  Argumentos de trabajo que se utilizan cuando se activa este desencadenador. En esta ejecución del trabajo, sustituyen a los argumentos predeterminados definidos en la propia definición del trabajo.

  Aquí puede especificar argumentos que su propio script de ejecución de tareas consume, así como argumentos que AWS Glue consume.

  Para obtener información acerca de cómo especificar y utilizar sus propios argumentos de trabajo, consulte Llamadas a las API de AWS Glue en Python en la guía para desarrolladores.

  Para obtener información acerca de los pares clave-valor que AWS Glue utiliza para configurar su trabajo, consulte el tema Parámetros especiales usados por AWS Glue en la guía para desarrolladores.
- `Timeout`: número (entero), como mínimo 1.

  Tiempo de espera de `JobRun` en minutos. Es el tiempo máximo que una ejecución de trabajo puede consumir recursos antes de que se termine y cambie al estado `TIMEOUT`. El valor predeterminado es 2880 minutos (48 horas). Sustituye el valor de tiempo de espera establecido en el flujo de trabajo principal.
- `SecurityConfiguration`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la estructura `SecurityConfiguration` que se va a utilizar con esta acción.
- `NotificationProperty`: un objeto NotificationProperty.

Especifica las propiedades de configuración de una notificación de ejecución de trabajo.

- `CrawlerName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre del rastreador que se va a utilizar con esta acción.

Estructura EventBatchingCondition

Condición del lote que debe cumplirse (número especificado de eventos recibidos o ventana de tiempo del lote vencida) antes de que se desencadene el evento EventBridge.

Campos

- `BatchSize` – Obligatorio: número (entero), que no es inferior a 1 ni es superior a 100.

  Número de eventos que se deben recibir de Amazon EventBridge antes de que se desencadene el evento EventBridge.

- `BatchWindow`: número (entero), que no es inferior a 1 ni es superior a 900.

  Ventana de tiempo en segundos después de la cual se activa el evento EventBridge. La ventana se inicia cuando se recibe el primer evento.

Operaciones

- Acción CreateTrigger (Python: create_trigger)
- Acción StartTrigger (Python: start_trigger)
- Acción GetTrigger (Python: get_trigger)
- Acción GetTriggers (Python: get_triggers)
- Acción UpdateTrigger (Python: update_trigger)
- Acción StopTrigger (Python: stop_trigger)
- Acción DeleteTrigger (Python: delete_trigger)
- Acción ListTriggers (Python: list_triggers)
- Acción BatchGetTriggers (Python: batch_get_triggers)

Acción CreateTrigger (Python: create_trigger)

Crea un nuevo disparador.

Solicitud

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre del desencadenador.

- `WorkflowName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre del flujo de trabajo asociado con el desencadenador.

- `Type` – Obligatorio: cadena UTF-8 (valores válidos: `SCHEDULED` | `CONDITIONAL` | `ON_DEMAND` | `EVENT`).

  Tipo del nuevo disparador.

- `Schedule`: cadena UTF-8.

  Expresión `cron` utilizada para especificar el programa (consulte Programaciones basadas en tiempo para trabajos y rastreadores). Por ejemplo, para ejecutar algo todos los días a las 12:15 UTC, especifique: `cron(15 12 * * ? *)`.

  Este campo es obligatorio cuando el tipo de disparador es SCHEDULED.

- `Predicate`: un objeto Predicate.

  Un predicado para especificar cuándo debe desencadenarse el nuevo disparador.

  Este campo es obligatorio cuando el tipo de disparador es `CONDITIONAL`.

- `Actions` (obligatorio): una matriz de objetos Acción.

  Acciones que inicia este disparador cuando se desencadena.

- `Description`: cadena de descripción de un máximo de 2048 bytes de largo, que coincide con el URI address multi-line string pattern.

  Descripción del nuevo disparador.

- `StartOnCreation`: booleano.

Establezca este parámetro en `true` para iniciar los disparadores `SCHEDULED` y `CONDITIONAL` cuando se crea. True no es compatible con los disparadores `ON_DEMAND`.

- `Tags`: matriz de mapas de pares de clave-valor, con 50 pares como máximo.

  Cada clave es una cadena UTF-8 con una longitud de entre 1 y 128 bytes.

  Cada valor es una cadena UTF-8 que no tiene más de 256 bytes de largo.

  Etiquetas que se van a utilizar con este desencadenador. Puede utilizar etiquetas para limitar el acceso al desencadenador. Para obtener más información acerca de las etiquetas en AWS Glue, consulte Etiquetas de AWS en AWS Glue en la guía para desarrolladores.

- `EventBatchingCondition`: un objeto EventBatchingCondition.

  Condición del lote que debe cumplirse (número especificado de eventos recibidos o ventana de tiempo del lote vencida) antes de que se desencadene el evento EventBridge.

Respuesta

- `Name`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre del desencadenador.

Errores

- `AlreadyExistsException`
- `EntityNotFoundException`
- `InvalidInputException`
- `IdempotentParameterMismatchException`
- `InternalServiceException`
- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`
- `ConcurrentModificationException`

Acción StartTrigger (Python: start_trigger)

Inicia un disparador ya existente. Consulte Activación de trabajos para obtener información acerca de cómo se desencadenan los diferentes tipos de disparadores.

Solicitud

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre del disparador que debe iniciarse.

Respuesta

- `Name`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre del disparador que se inició.

Errores

- `InvalidInputException`
- `InternalServiceException`
- `EntityNotFoundException`
- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`
- `ConcurrentRunsExceededException`

Acción GetTrigger (Python: get_trigger)

Recupera la definición de un disparador.

Solicitud

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre del disparador que debe recuperarse.

Respuesta

- `Trigger`: un objeto Desencadenador.

  Definición del disparador solicitada.

Errores

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

Acción GetTriggers (Python: get_triggers)

Obtiene todos los disparadores asociados a un flujo de trabajo.

Solicitud

- `NextToken`: cadena UTF-8.

  Token de continuación si se trata de una llamada de continuidad.
- `DependentJobName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre del trabajo para recuperar disparadores. Se devolverá el disparador que puede iniciar este trabajo y, si dicho disparador no existe, se devolverán todos los disparadores.
- `MaxResults`: número (entero), mayor que 1 y menor que 200.

  Tamaño máximo de la respuesta.

Respuesta

- `Triggers`: matriz de objetos Desencadenador.

  Lista de disparadores para el flujo de trabajo especificado.
- `NextToken`: cadena UTF-8.

Token de continuación si todavía no se han devuelto todos los disparadores solicitados.

Errores

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

Acción UpdateTrigger (Python: update_trigger)

Actualiza una definición de disparador.

Solicitud

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre del disparador que debe actualizarse.
- `TriggerUpdate`: obligatorio: objeto TriggerUpdate.

  Nuevos valores con los que se actualizará el disparador.

Respuesta

- `Trigger`: un objeto Desencadenador.

  Definición del disparador resultante.

Errores

- `InvalidInputException`
- `InternalServiceException`
- `EntityNotFoundException`
- `OperationTimeoutException`
- `ConcurrentModificationException`

Acción StopTrigger (Python: stop_trigger)

Detiene un disparador especificado.

Solicitud

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre del disparador que debe detenerse.

Respuesta

- `Name`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre del disparador que se detuvo.

Errores

- `InvalidInputException`
- `InternalServiceException`
- `EntityNotFoundException`
- `OperationTimeoutException`
- `ConcurrentModificationException`

Acción DeleteTrigger (Python: delete_trigger)

Elimina un disparador especificado. Si no se encuentra el disparador, no se genera una excepción.

Solicitud

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre del disparador que se va a eliminar.

Respuesta

- `Name`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre del disparador que se eliminó.

Errores

- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `ConcurrentModificationException`

Acción ListTriggers (Python: list_triggers)

Recupera los nombres de todos los recursos de desencadenadores de esta cuenta de AWS o los recursos con la etiqueta especificada. Esta operación permite ver qué recursos están disponibles en la cuenta y sus nombres.

Esta operación toma el campo `Tags` opcional, que se puede utilizar como filtro en la respuesta para que los recursos etiquetados se devuelvan agrupados. Si decide utilizar el filtrado de etiquetas, solo se devolverán los recursos con la etiqueta especificada.

Solicitud

- `NextToken`: cadena UTF-8.

  Token de continuación, si se trata de una solicitud de continuidad.
- `DependentJobName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre del trabajo para el que se quieren recuperar disparadores. Se devuelve el disparador que puede comenzar este trabajo. Si no existe dicho disparador, se devuelven todos los disparadores.
- `MaxResults`: número (entero), mayor que 1 y menor que 200.

  Tamaño máximo de una lista que se devolverá.
- `Tags`: matriz de mapas de pares de clave-valor, con 50 pares como máximo.

Cada clave es una cadena UTF-8 con una longitud de entre 1 y 128 bytes.

Cada valor es una cadena UTF-8 que no tiene más de 256 bytes de largo.

Especifica que se devuelvan solamente los recursos etiquetados.

Respuesta

- `TriggerNames`: matriz de cadenas UTF-8.

  Nombres de todos los disparadores de la cuenta o de los disparadores con las etiquetas especificadas.
- `NextToken`: cadena UTF-8.

  Token de continuación, si la lista devuelta no contiene la última métrica disponible.

Errores

- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

Acción BatchGetTriggers (Python: batch_get_triggers)

Devuelve la lista de metadatos de recursos de una determinada lista de nombres de disparadores. Después de llamar a la operación `ListTriggers`, puede llamar a esta operación para obtener acceso a los datos a los que ha concedido permisos. Esta operación admite todos los permisos de IAM, incluidas las condiciones de permisos que utilizan etiquetas.

Solicitud

- `TriggerNames` – Obligatorio: una matriz de cadenas UTF-8.

  Lista de nombres de disparadores, que pueden ser los nombres devueltos en la operación `ListTriggers`.

Respuesta

- `Triggers`: matriz de objetos Desencadenador.

  Lista de definiciones de disparadores.

- `TriggersNotFound`: matriz de cadenas UTF-8.

  No se encuentra ninguna lista de nombres de disparadores.

Errores

- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`

API de sesiones interactivas

La API de sesiones interactivas describe la AWS Glue API relacionada con el uso de sesiones AWS Glue interactivas para crear y probar scripts de extracción, transformación y carga (ETL) para la integración de datos.

Tipos de datos

- Estructura de sesión
- SessionCommand estructura
- Estructura de instrucción
- StatementOutput estructura
- StatementOutputData estructura
- ConnectionsList estructura

Estructura de sesión

El período en el que se ejecuta un entorno en tiempo de ejecución de Spark remoto.

Campos

- `Id`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID de la sesión.

- `CreatedOn`: marca temporal.

  La fecha y hora en que se creó la sesión.

- `Status`: cadena UTF-8 (valores válidos: `PROVISIONING` | `READY` | `FAILED` | `TIMEOUT` | `STOPPING` | `STOPPED`).

  El estado de la sesión.

- `ErrorMessage`: cadena de descripción de un máximo de 2048 bytes de largo, que coincide con el URI address multi-line string pattern.

  El mensaje de error que se muestra durante la sesión.

- `Description`: cadena de descripción de un máximo de 2048 bytes de largo, que coincide con el URI address multi-line string pattern.

  La descripción de la sesión.

- `Role`: cadena UTF-8, con 20 bytes como mínimo o más de 2048 bytes de largo, que coincide con el Custom string pattern #26.

  El nombre o nombre de recurso de Amazon (ARN) del rol de IAM asociado a la sesión.

- `Command`: un objeto SessionCommand.

  El comando Object.see. SessionCommand

- `DefaultArguments`: matriz de asignación de pares de clave-valor, con 75 pares como máximo.

  Cada clave es una cadena UTF-8 con 1 byte de largo como mínimo y 128 bytes de largo como máximo, que coincide con Custom string pattern #27.

  Cada valor es una cadena UTF-8, con 4096 bytes de largo como máximo, que coincide con URI address multi-line string pattern.

  Una matriz de mapas de pares clave-valor. El máximo es de 75 pares.

- `Connections`: un objeto ConnectionsList.

El número de conexiones utilizadas para la sesión.

- `Progress`: número (doble).

  El progreso de la ejecución del código de la sesión.

- `MaxCapacity`: número (doble).

  El número de unidades de procesamiento de AWS Glue datos (DPUs) que se pueden asignar cuando se ejecuta el trabajo. Una DPU es una medida relativa de la potencia de procesamiento que consta de 4 vCPU de capacidad de computación y 16 GB de memoria.

- `SecurityConfiguration`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la SecurityConfiguration estructura que se va a utilizar con la sesión.

- `GlueVersion`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Custom string pattern #20.

  La AWS Glue versión determina las versiones de Apache Spark y Python AWS Glue compatibles. GlueVersion Debe ser superior a la 2.0.

- `DataAccessId`: cadena UTF-8, con 1 byte como mínimo o más de 36 bytes de largo.

  El ID de acceso a datos de la sesión.

- `PartitionId`: cadena UTF-8, con 1 byte como mínimo o más de 36 bytes de largo.

  El ID de partición de la sesión.

- `NumberOfWorkers`: número (entero).

  El número de trabajadores de una `WorkerType` definida para usar para la sesión.

- `WorkerType`: cadena UTF-8 (valores válidos: `Standard=""` | `G.1X=""` | `G.2X=""` | `G.025X=""` | `G.4X=""` | `G.8X=""` | `Z.2X=""`).

  El tipo de trabajador predefinido que se asigna cuando se ejecuta una sesión. Acepta un valor de `G.1X`, `G.2X`, `G.4X` o `G.8X` para las sesiones de Spark. Acepta el valor `Z.2X` para las sesiones de Ray.

- `CompletedOn`: marca temporal.

  La fecha y la hora en que se completó la ejecución de esta sesión.

- `ExecutionTime`: número (doble).

El tiempo total de duración de la sesión.

- `DPUSeconds`: número (doble).

  Las DPU consumidas por la sesión (fórmula: ExecutionTime * MaxCapacity).

- `IdleTimeout`: número (entero).

  Número de minutos sin actividad antes de que se agote el tiempo de espera de la sesión.

- `ProfileName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de un perfil AWS Glue de uso asociado a la sesión.

SessionCommand estructura

El `SessionCommand` que ejecuta este trabajo.

Campos

- `Name`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Especifica el nombre del SessionCommand. Puede ser “glueetl” o “gluestreaming”.

- `PythonVersion`: cadena UTF-8 que coincide con el Custom string pattern #21.

  Especifica la versión de Python. La versión de Python indica la versión admitida para trabajos de tipo Spark.

Estructura de instrucción

La instrucción o solicitud de que se produzca una acción concreta en una sesión.

Campos

- `Id`: número (entero).

  El ID de la instrucción.

- `Code`: cadena UTF-8.

El código de ejecución de la instrucción.

- `State`: cadena UTF-8 (valores válidos: `WAITING` | `RUNNING` | `AVAILABLE` | `CANCELLING` | `CANCELLED` | `ERROR`).

  El estado mientras se ejecuta la solicitud.

- `Output`: un objeto StatementOutput.

  El resultado en JSON.

- `Progress`: número (doble).

  El progreso de ejecución del código.

- `StartedOn`: número (largo).

  La fecha y hora unix en que se inició la definición del trabajo.

- `CompletedOn`: número (largo).

  La fecha y hora unix en que se completó la definición del trabajo.

StatementOutput estructura

El resultado de la ejecución de código en formato JSON.

Campos

- `Data`: un objeto StatementOutputData.

  El resultado de la ejecución de código.

- `ExecutionCount`: número (entero).

  El recuento de ejecución del resultado.

- `Status`: cadena UTF-8 (valores válidos: `WAITING` | `RUNNING` | `AVAILABLE` | `CANCELLING` | `CANCELLED` | `ERROR`).

  El estado del resultado de ejecución de código.

- `ErrorName`: cadena UTF-8.

  El nombre del error en el resultado.

- `ErrorValue`: cadena UTF-8.

El valor de error del resultado.

- `Traceback`: matriz de cadenas UTF-8.

  El seguimiento del resultado.

StatementOutputData estructura

El resultado de la ejecución de código en formato JSON.

Campos

- `TextPlain`: cadena UTF-8.

  El resultado de la ejecución de código en formato de texto.

ConnectionsList estructura

Especifica las conexiones que utiliza un flujo de trabajo.

Campos

- `Connections`: matriz de cadenas UTF-8.

  Lista de conexiones que utiliza el flujo de trabajo.

Operaciones

- CreateSession acción (Python: create_session)
- StopSession acción (Python: stop_session)
- DeleteSession acción (Python: delete_session)
- GetSession acción (Python: get_session)
- ListSessions acción (Python: list_sessions)
- RunStatement acción (Python: run_statement)
- CancelStatement acción (Python: cancel_statement)
- GetStatement acción (Python: get_statement)
- ListStatements acción (Python: list_statements)

CreateSession acción (Python: create_session)

Crea una nueva sesión.

Solicitud

Solicitud para crear una nueva sesión.

- `Id`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID de la solicitud de sesión.
- `Description`: cadena de descripción de un máximo de 2048 bytes de largo, que coincide con el URI address multi-line string pattern.

  La descripción de la sesión.
- `Role`: obligatorio: cadena UTF-8, con 20 bytes de largo o más de 2048 bytes de largo, que coincide con el Custom string pattern #26.

  El ARN del rol de IAM
- `Command`: obligatorio: objeto SessionCommand.

  El `SessionCommand` que pone en marcha el trabajo.
- `Timeout`: número (entero), como mínimo 1.

  Número de minutos que deben transcurrir para que se agote el tiempo de espera de la sesión. El valor predeterminado para los trabajos de ETL de Spark es de 48 horas (2880 minutos): la duración máxima de la sesión para este tipo de trabajo. Consulte la documentación para otros tipos de trabajo.
- `IdleTimeout`: número (entero), como mínimo 1.

  Número de minutos sin actividad que deben transcurrir para que se agote el tiempo de espera de la sesión. El valor predeterminado para los trabajos de ETL de Spark es el valor de Tiempo de espera. Consulte la documentación para otros tipos de trabajo.
- `DefaultArguments`: matriz de asignación de pares de clave-valor, con 75 pares como máximo.

  Cada clave es una cadena UTF-8 con 1 byte de largo como mínimo y 128 bytes de largo como máximo, que coincide con Custom string pattern #27.

Cada valor es una cadena UTF-8, con 4096 bytes de largo como máximo, que coincide con URI address multi-line string pattern.

Una matriz de mapas de pares clave-valor. El máximo es de 75 pares.

- `Connections`: un objeto ConnectionsList.

  Número de conexiones que se van a utilizar para la sesión.

- `MaxCapacity`: número (doble).

  El número de unidades de procesamiento de AWS Glue datos (DPUs) que se pueden asignar cuando se ejecuta el trabajo. Una DPU es una medida relativa de la potencia de procesamiento que consta de 4 vCPU de capacidad de computación y 16 GB de memoria.

- `NumberOfWorkers`: número (entero).

  El número de trabajadores de una `WorkerType` definida para usar para la sesión.

- `WorkerType`: cadena UTF-8 (valores válidos: `Standard=""` | `G.1X=""` | `G.2X=""` | `G.025X=""` | `G.4X=""` | `G.8X=""` | `Z.2X=""`).

  El tipo de proceso de trabajo predefinido que se asigna cuando se ejecuta un trabajo. Acepta un valor de G.1X, G.2X, G.4X o G.8X para los trabajos de Spark. Acepta el valor Z.2X para los portátiles Ray.

  - Para el tipo de trabajador `G.1X`, cada trabajador se asocia a 1 DPU (4 vCPU, 16 GB de memoria) con un disco de 84 GB (aproximadamente 34 GB libres) y proporciona 1 ejecutor por trabajador. Recomendamos este tipo de trabajador para cargas de trabajo como transformaciones de datos, uniones y consultas, ya que ofrece una forma escalable y rentable de ejecutar la mayoría de los trabajos.
  - Para el tipo de trabajador `G.2X`, cada trabajador se asocia a 2 DPU (8 vCPU, 32 GB de memoria) con un disco de 128 GB (aproximadamente 77 GB libres) y proporciona 1 ejecutor por trabajador. Recomendamos este tipo de trabajador para cargas de trabajo como transformaciones de datos, uniones y consultas, ya que ofrece una forma escalable y rentable de ejecutar la mayoría de los trabajos.
  - Para el tipo de trabajador `G.4X`, cada trabajador se asocia a 4 DPU (16 vCPU, 64 GB de memoria) con un disco de 256 GB (aproximadamente 235 GB libres) y proporciona 1 ejecutor por trabajador. Recomendamos este tipo de trabajador para los trabajos cuyas cargas de trabajo contengan las transformaciones, agregaciones, uniones y consultas más exigentes. Este tipo de trabajador solo está disponible para los trabajos de Spark ETL de la AWS Glue versión 3.0

o posteriores en AWS las siguientes regiones: EE.UU. Este (Ohio), EE.UU. Este (Norte de Virginia), EE.UU. Oeste (Oregón), Asia Pacífico (Singapur), Asia Pacífico (Sídney), Asia Pacífico (Tokio), Canadá (Central), Europa (Fráncfort), Europa (Irlanda) y Europa (Estocolmo).

- Para el tipo de trabajador `G.8X`, cada trabajador se asocia a 8 DPU (32 GB vCPU, 128 GB de memoria) con un disco de 512 GB (aproximadamente 487 GB libres) y proporciona 1 ejecutor por trabajador. Recomendamos este tipo de trabajador para los trabajos cuyas cargas de trabajo contengan las transformaciones, agregaciones, uniones y consultas más exigentes. Este tipo de trabajador solo está disponible para los trabajos ETL de Spark de la AWS Glue versión 3.0 o posteriores, en las mismas AWS regiones compatibles con el tipo de `G.4X` trabajador.
- Para el tipo de trabajador `Z.2X`, cada trabajador se asigna a 2 M-DPU (8 vCPU, 64 GB de memoria) con un disco de 128 GB (aproximadamente 120 GB libres) y proporciona hasta 8 trabajadores de Ray en función del escalador automático.

- `SecurityConfiguration`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la SecurityConfiguration estructura que se utilizará con la sesión

- `GlueVersion`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Custom string pattern #20.

  La AWS Glue versión determina las versiones de Apache Spark y Python AWS Glue compatibles. GlueVersion Debe ser superior a la 2.0.

- `DataAccessId`: cadena UTF-8, con 1 byte como mínimo o más de 36 bytes de largo.

  El ID de acceso a datos de la sesión.

- `PartitionId`: cadena UTF-8, con 1 byte como mínimo o más de 36 bytes de largo.

  El ID de partición de la sesión.

- `Tags`: matriz de mapas de pares de clave-valor, con 50 pares como máximo.

  Cada clave es una cadena UTF-8 con una longitud de entre 1 y 128 bytes.

  Cada valor es una cadena UTF-8 que no tiene más de 256 bytes de largo.

  El mapa de pares de valor de clave (etiquetas) pertenecientes a la sesión.

- `RequestOrigin`: cadena UTF-8, con 1 byte de largo como mínimo y 128 bytes de largo como máximo, que coincide con el Custom string pattern #27.

El origen de la solicitud.

- `ProfileName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de un perfil de AWS Glue uso asociado a la sesión.

Respuesta

- `Session`: un objeto Sesión.

  Arroja el objeto de la sesión en la respuesta.

Errores

- `AccessDeniedException`
- `IdempotentParameterMismatchException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`
- `ValidationException`
- `AlreadyExistsException`
- `ResourceNumberLimitExceededException`

StopSession acción (Python: stop_session)

Detiene la sesión.

Solicitud

- `Id`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID de la sesión que se va a detener.
- `RequestOrigin`: cadena UTF-8, con 1 byte de largo como mínimo y 128 bytes de largo como máximo, que coincide con el Custom string pattern #27.

El origen de la solicitud.

Respuesta

- `Id`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Arroja el ID de la sesión detenida.

Errores

- `AccessDeniedException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`
- `IllegalSessionStateException`
- `ConcurrentModificationException`

DeleteSession acción (Python: delete_session)

Elimina la sesión.

Solicitud

- `Id`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID de la sesión que se va a eliminar.
- `RequestOrigin`: cadena UTF-8, con 1 byte de largo como mínimo y 128 bytes de largo como máximo, que coincide con el Custom string pattern #27.

  El nombre del origen de la solicitud de eliminación de sesión.

Respuesta

- `Id`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Arroja el ID de la sesión eliminada.

Errores

- `AccessDeniedException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`
- `IllegalSessionStateException`
- `ConcurrentModificationException`

GetSession acción (Python: get_session)

Recupera la sesión.

Solicitud

- `Id`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Token de continuación si todavía no se arrojaron todas las instrucciones.
- `RequestOrigin`: cadena UTF-8, con 1 byte de largo como mínimo y 128 bytes de largo como máximo, que coincide con el Custom string pattern #27.

  El origen de la solicitud.

Respuesta

- `Session`: un objeto Sesión.

  Se recupera el objeto de la sesión en la respuesta.

Errores

- `AccessDeniedException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`

ListSessions acción (Python: list_sessions)

Recupere una lista de sesiones.

Solicitud

- `NextToken`: cadena UTF-8 de 400 000 bytes de largo como máximo.

  El token para el próximo conjunto de resultados o nulo si no hay más resultados.
- `MaxResults`: número (entero) que no es inferior a 1 ni es superior a 1000.

  El número máximo de resultados.
- `Tags`: matriz de mapas de pares de clave-valor, con 50 pares como máximo.

  Cada clave es una cadena UTF-8 con una longitud de entre 1 y 128 bytes.

  Cada valor es una cadena UTF-8 que no tiene más de 256 bytes de largo.

  Etiquetas pertenecientes a la sesión.
- `RequestOrigin`: cadena UTF-8, con 1 byte de largo como mínimo y 128 bytes de largo como máximo, que coincide con el Custom string pattern #27.

  El origen de la solicitud.

Respuesta

- `Ids`: matriz de cadenas UTF-8.

  Arroja el ID de la sesión.
- `Sessions`: matriz de objetos Sesión.

Arroja el objeto de la sesión.

- `NextToken`: cadena UTF-8 de 400 000 bytes de largo como máximo.

  El token para el próximo conjunto de resultados o nulo si no hay más resultados.

Errores

- `AccessDeniedException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

RunStatement acción (Python: run_statement)

Pone en marcha la instrucción.

Solicitud

- `SessionId`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID de sesión de la instrucción que se ejecutará.
- `Code`: obligatorio: cadena UTF-8, de 68 000 bytes de largo como máximo.

  El código de instrucción que se pondrá en marcha.
- `RequestOrigin`: cadena UTF-8, con 1 byte de largo como mínimo y 128 bytes de largo como máximo, que coincide con el Custom string pattern #27.

  El origen de la solicitud.

Respuesta

- `Id`: número (entero).

  Arroja el ID de la instrucción que se puso en marcha.

Errores

- `EntityNotFoundException`
- `AccessDeniedException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`
- `ValidationException`
- `ResourceNumberLimitExceededException`
- `IllegalSessionStateException`

CancelStatement acción (Python: cancel_statement)

Cancela la instrucción.

Solicitud

- `SessionId`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID de sesión de la instrucción que se va a cancelar.
- `Id` – Obligatorio: número (entero).

  El ID de la instrucción que se va a cancelar.
- `RequestOrigin`: cadena UTF-8, con 1 byte de largo como mínimo y 128 bytes de largo como máximo, que coincide con el Custom string pattern #27.

  El origen de la solicitud de cancelación de la instrucción.

Respuesta

- Sin parámetros de respuesta.

Errores

- `AccessDeniedException`

- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`
- `IllegalSessionStateException`

GetStatement acción (Python: get_statement)

Recupera la instrucción.

Solicitud

- `SessionId`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID de sesión de la instrucción.
- `Id` – Obligatorio: número (entero).

  El ID de la instrucción.
- `RequestOrigin`: cadena UTF-8, con 1 byte de largo como mínimo y 128 bytes de largo como máximo, que coincide con el Custom string pattern #27.

  El origen de la solicitud.

Respuesta

- `Statement`: un objeto Instrucción.

  Arroja la instrucción.

Errores

- `AccessDeniedException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`

- `InvalidInputException`
- `IllegalSessionStateException`

ListStatements acción (Python: list_statements)

Muestra las instrucciones de la sesión.

Solicitud

- `SessionId`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID de sesión de las instrucciones.
- `RequestOrigin`: cadena UTF-8, con 1 byte de largo como mínimo y 128 bytes de largo como máximo, que coincide con el Custom string pattern #27.

  El origen de la solicitud de enumeración de instrucciones.
- `NextToken`: cadena UTF-8 de 400 000 bytes de largo como máximo.

  Token de continuación si se trata de una llamada de continuidad.

Respuesta

- `Statements`: matriz de objetos Instrucción.

  Arroja la lista de instrucciones.
- `NextToken`: cadena UTF-8 de 400 000 bytes de largo como máximo.

  Token de continuación si todavía no se arrojaron todas las instrucciones.

Errores

- `AccessDeniedException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`

- `IllegalSessionStateException`

API de puntos de conexión de desarrollo

La API de puntos de conexión de desarrollo describe la API de AWS Glue relacionada con las pruebas mediante un DevEndpoint (punto de conexión de desarrollo) personalizado.

Tipos de datos

- Estructura DevEndpoint
- Estructura DevEndpointCustomLibraries

Estructura DevEndpoint

Un punto de enlace de desarrollo en el que un desarrollador pueda depurar de forma remota scripts de extracción, transformación y carga (ETL).

Campos

- `EndpointName`: cadena UTF-8.

  Nombre del elemento `DevEndpoint`.
- `RoleArn`: cadena UTF-8 que coincide con el AWS IAM ARN string pattern.

  El nombre de recurso de Amazon (ARN) del rol de IAM que se utiliza en este `DevEndpoint`.
- `SecurityGroupIds`: matriz de cadenas UTF-8.

  Una lista de identificadores del grupo de seguridad que se utilizan en este `DevEndpoint`.
- `SubnetId`: cadena UTF-8.

  El ID de subred para este `DevEndpoint`.
- `YarnEndpointAddress`: cadena UTF-8.

  La dirección del punto de enlace de YARN que utiliza este `DevEndpoint`.
- `PrivateAddress`: cadena UTF-8.

Una dirección IP privada para acceder al `DevEndpoint` dentro de una VPC si los datos `DevEndpoint` se crean dentro de una. El campo `PrivateAddress` está presente solo cuando se crea el `DevEndpoint` dentro de la VPC.

- `ZeppelinRemoteSparkInterpreterPort`: número (entero).

  El puerto de Apache Zeppelin para el intérprete remoto de Apache Spark.

- `PublicAddress`: cadena UTF-8.

  La dirección IP pública que utiliza este `DevEndpoint`. El campo `PublicAddress` está presente solo cuando se crea un `DevEndpoint` no de nube privada virtual (VPC).

- `Status`: cadena UTF-8.

  El estado actual de este `DevEndpoint`.

- `WorkerType`: cadena UTF-8 (valores válidos: `Standard=""` | `G.1X=""` | `G.2X=""` | `G.025X=""` | `G.4X=""` | `G.8X=""` | `Z.2X=""`).

  El tipo de proceso de trabajo predefinido que se asigna al punto de enlace de desarrollo. Admite un valor de Standard, G.1X o G.2X.

  - Para el tipo de proceso de trabajo `Standard`, cada proceso de trabajo proporciona 4 vCPU, 16 GB de memoria y un disco de 50 GB y 2 ejecutores por trabajador.
  - Para el tipo de proceso de trabajo `G.1X`, cada proceso de trabajo se asocia a 1 DPU (4 vCPU, 16 GB de memoria, disco de 64 GB) y proporciona 1 ejecutor por proceso de trabajo. Le recomendamos este tipo de proceso de trabajo para trabajos con un uso intensivo de la memoria.
  - Para el tipo de proceso de trabajo `G.2X`, cada proceso de trabajo se asocia a 2 DPU (8 vCPU, 32 GB de memoria, disco de 128 GB) y proporciona 1 ejecutor por proceso de trabajo. Le recomendamos este tipo de proceso de trabajo para trabajos con un uso intensivo de la memoria.

  Problema conocido: cuando se crea un punto de enlace de desarrollo con la configuración `WorkerType` `G.2X`, los controladores Spark para el punto de enlace de desarrollo se ejecutarán en 4 vCPU, 16 GB de memoria y un disco de 64 GB.

- `GlueVersion`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Custom string pattern #20.

La versión de Glue determina las versiones de Apache Spark y Python que admite AWS Glue. La versión de Python indica la versión admitida para ejecutar sus scripts de ETL en puntos de enlace de desarrollo.

Para obtener más información acerca de las versiones de AWS Glue disponibles y las versiones de Spark y Python correspondientes, consulte Versión de Glue en la guía para desarrolladores.

Los puntos de enlace de desarrollo que se crean sin especificar una versión de Glue se establecen de forma predeterminada en Glue 0.9.

Puede especificar una versión de Python compatible con puntos de enlace de desarrollo mediante el parámetro `Arguments` en las API `UpdateDevEndpoint` o `CreateDevEndpoint`. Si no se proporcionan argumentos, la versión es Python 2 de forma predeterminada.

- `NumberOfWorkers`: número (entero).

  El número de procesos de trabajo de un definido `workerType` que se asignan al punto de enlace de desarrollo.

  El número máximo de procesos de trabajo que puede definir son 299 para `G.1X` y 149 para `G.2X`.

- `NumberOfNodes`: número (entero).

  El número de unidades de procesamiento de datos (DPU) de AWS Glue asignadas a este `DevEndpoint`.

- `AvailabilityZone`: cadena UTF-8.

  La zona de disponibilidad de AWS en la que se encuentra este `DevEndpoint`.

- `VpcId`: cadena UTF-8.

  El ID de la nube privada virtual (VPC) que utiliza este `DevEndpoint`.

- `ExtraPythonLibsS3Path`: cadena UTF-8.

  Las rutas a una o varias bibliotecas Python en un bucket de Amazon S3 que debe estar cargado en su `DevEndpoint`. Varios valores deben ser rutas completas separadas por una coma.

> **Note**
>
> Solo puede utilizar bibliotecas de Python puras con un `DevEndpoint`. Las bibliotecas que se basan en extensiones de C, como la biblioteca de análisis de datos de Python pandas, no se admiten actualmente.

- `ExtraJarsS3Path`: cadena UTF-8.

  La ruta a uno o más archivos `.jar` de Java en un bucket de S3 que se debe cargar en su `DevEndpoint`.

> **Note**
>
> Solo puede utilizar bibliotecas Java/Scala puras con un `DevEndpoint`.

- `FailureReason`: cadena UTF-8.

  El motivo de un error actual en este `DevEndpoint`.

- `LastUpdateStatus`: cadena UTF-8.

  El estado de la última actualización.

- `CreatedTimestamp`: marca temporal.

  El momento en que se creó este DevEndpoint.

- `LastModifiedTimestamp`: marca temporal.

  El punto en el que este `DevEndpoint` se modificó por última vez.

- `PublicKey`: cadena UTF-8.

  La clave pública que va a utilizar este `DevEndpoint` para autenticación. Este atributo se proporciona para compatibilidad con versiones anteriores, ya que el atributo que se recomienda usar son las claves públicas.

- `PublicKeys`: matriz de cadenas UTF-8, con 5 cadenas como máximo.

  Una lista de claves públicas que van a utilizar los `DevEndpoints` para autenticación. Se prefiere el uso de este atributo a una sola clave pública, ya que las claves públicas le permiten tener una clave privada diferente por cliente.

> **Note**
>
> Si ha creado previamente un punto de enlace con una clave pública, debe eliminar dicha clave para poder establecer una lista de claves públicas. Realice una llamada a la operación API `UpdateDevEndpoint` con el contenido de la clave pública en el atributo `deletePublicKeys` y la lista de nuevas claves en el atributo `addPublicKeys`.

- `SecurityConfiguration`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la estructura `SecurityConfiguration` que se va a utilizar con este `DevEndpoint`.

- `Arguments`: matriz de mapas de pares de clave-valor, con 100 pares como máximo.

  Cada clave es una cadena UTF-8.

  Cada valor es una cadena UTF-8.

  Mapa de argumentos que se utiliza para configurar `DevEndpoint`.

  Los argumentos válidos son:

  - `"--enable-glue-datacatalog": ""`

  Puede especificar una versión de Python compatible con puntos de enlace de desarrollo mediante el parámetro `Arguments` en las API `UpdateDevEndpoint` o `CreateDevEndpoint`. Si no se proporcionan argumentos, la versión es Python 2 de forma predeterminada.

Estructura DevEndpointCustomLibraries

Las bibliotecas personalizadas que se van a cargar en un punto de enlace.

Campos

- `ExtraPythonLibsS3Path`: cadena UTF-8.

  Las rutas a una o varias bibliotecas Python en un bucket de Amazon Simple Storage Service (Amazon S3) que deben cargarse en el `DevEndpoint`. Varios valores deben ser rutas completas separadas por una coma.

> **Note**
>
> Solo puede utilizar bibliotecas de Python puras con un `DevEndpoint`. Las bibliotecas que se basan en extensiones de C, como la biblioteca de análisis de datos de Python pandas, no se admiten actualmente.

- `ExtraJarsS3Path`: cadena UTF-8.

  La ruta a uno o más archivos `.jar` de Java en un bucket de S3 que se debe cargar en su `DevEndpoint`.

> **Note**
>
> Solo puede utilizar bibliotecas Java/Scala puras con un `DevEndpoint`.

Operaciones

- Acción CreateDevEndpoint (Python: create_dev_endpoint)
- Acción UpdateDevEndpoint (Python: update_dev_endpoint)
- Acción DeleteDevEndpoint (Python: delete_dev_endpoint)
- Acción GetDevEndpoint (Python: get_dev_endpoint)
- Acción GetDevEndpoints (Python: get_dev_endpoints)
- Acción BatchGetDevEndpoints (Python: batch_get_dev_endpoints)
- Acción ListDevEndpoints (Python: list_dev_endpoints)

Acción CreateDevEndpoint (Python: create_dev_endpoint)

Crea un nuevo punto de enlace de desarrollo.

Solicitud

- `EndpointName` – Obligatorio: cadena UTF-8.

  El nombre que se va a asignar al nuevo `DevEndpoint`.
- `RoleArn`: obligatorio: cadena UTF-8 que coincide con el AWS IAM ARN string pattern.

El rol de IAM para el `DevEndpoint`.

- `SecurityGroupIds`: matriz de cadenas UTF-8.

  Los ID de grupo de seguridad para los grupos de seguridad que va a utilizar el nuevo `DevEndpoint`.

- `SubnetId`: cadena UTF-8.

  El ID de subred para el nuevo `DevEndpoint` que se va a utilizar.

- `PublicKey`: cadena UTF-8.

  La clave pública que va a utilizar este `DevEndpoint` para autenticación. Este atributo se proporciona para compatibilidad con versiones anteriores, ya que el atributo que se recomienda usar son las claves públicas.

- `PublicKeys`: matriz de cadenas UTF-8, con 5 cadenas como máximo.

  Lista de claves públicas que van a utilizar los puntos de enlace de desarrollo para la autenticación. Se prefiere el uso de este atributo a una sola clave pública, ya que las claves públicas le permiten tener una clave privada diferente por cliente.

  > **Note**
  >
  > Si ha creado previamente un punto de enlace con una clave pública, debe eliminar dicha clave para poder establecer una lista de claves públicas. Llame a la API `UpdateDevEndpoint` con el contenido de la clave pública en el atributo `deletePublicKeys` y la lista de nuevas claves en el atributo `addPublicKeys`.

- `NumberOfNodes`: número (entero).

  El número de unidades de procesamiento de datos (DPU) de AWS Glue que se van a asignar a este `DevEndpoint`.

- `WorkerType`: cadena UTF-8 (valores válidos: `Standard=""` | `G.1X=""` | `G.2X=""` | `G.025X=""` | `G.4X=""` | `G.8X=""` | `Z.2X=""`).

  El tipo de proceso de trabajo predefinido que se asigna al punto de enlace de desarrollo. Admite un valor de Standard, G.1X o G.2X.

  - Para el tipo de proceso de trabajo `Standard`, cada proceso de trabajo proporciona 4 vCPU, 16 GB de memoria y un disco de 50 GB y 2 ejecutores por trabajador.

- Para el tipo de proceso de trabajo `G.1X`, cada proceso de trabajo se asocia a 1 DPU (4 vCPU, 16 GB de memoria, disco de 64 GB) y proporciona 1 ejecutor por proceso de trabajo. Le recomendamos este tipo de proceso de trabajo para trabajos con un uso intensivo de la memoria.
- Para el tipo de proceso de trabajo `G.2X`, cada proceso de trabajo se asocia a 2 DPU (8 vCPU, 32 GB de memoria, disco de 128 GB) y proporciona 1 ejecutor por proceso de trabajo. Le recomendamos este tipo de proceso de trabajo para trabajos con un uso intensivo de la memoria.

Problema conocido: cuando se crea un punto de enlace de desarrollo con la configuración `WorkerType` `G.2X`, los controladores Spark para el punto de enlace de desarrollo se ejecutarán en 4 vCPU, 16 GB de memoria y un disco de 64 GB.

- `GlueVersion`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Custom string pattern #20.

  La versión de Glue determina las versiones de Apache Spark y Python que admite AWS Glue. La versión de Python indica la versión admitida para ejecutar sus scripts de ETL en puntos de enlace de desarrollo.

  Para obtener más información acerca de las versiones de AWS Glue disponibles y las versiones de Spark y Python correspondientes, consulte Versión de Glue en la guía para desarrolladores.

  Los puntos de enlace de desarrollo que se crean sin especificar una versión de Glue se establecen de forma predeterminada en Glue 0.9.

  Puede especificar una versión de Python compatible con puntos de enlace de desarrollo mediante el parámetro `Arguments` en las API `UpdateDevEndpoint` o `CreateDevEndpoint`. Si no se proporcionan argumentos, la versión es Python 2 de forma predeterminada.

- `NumberOfWorkers`: número (entero).

  El número de procesos de trabajo de un definido `workerType` que se asignan al punto de enlace de desarrollo.

  El número máximo de procesos de trabajo que puede definir son 299 para `G.1X` y 149 para `G.2X`.

- `ExtraPythonLibsS3Path`: cadena UTF-8.

  Las rutas a una o varias bibliotecas Python en un bucket de Amazon S3 que debe estar cargado en su `DevEndpoint`. Varios valores deben ser rutas completas separadas por una coma.

> **Note**
>
> Solo puede utilizar bibliotecas de Python puras con un `DevEndpoint`. Todavía no se admiten las bibliotecas que se basan en las extensiones de C, como la biblioteca de análisis de datos Python pandas.

- `ExtraJarsS3Path`: cadena UTF-8.

  La ruta a uno o más archivos `.jar` de Java en un bucket de S3 que se debe cargar en su `DevEndpoint`.

- `SecurityConfiguration`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la estructura `SecurityConfiguration` que se va a utilizar con este `DevEndpoint`.

- `Tags`: matriz de mapas de pares de clave-valor, con 50 pares como máximo.

  Cada clave es una cadena UTF-8 con una longitud de entre 1 y 128 bytes.

  Cada valor es una cadena UTF-8 que no tiene más de 256 bytes de largo.

  Etiquetas que se van a utilizar con este objeto DevEndpoint. Puede utilizar etiquetas para limitar el acceso al objeto DevEndpoint. Para obtener más información acerca de las etiquetas en AWS Glue, consulte Etiquetas de AWS en AWS Glue en la guía para desarrolladores.

- `Arguments`: matriz de mapas de pares de clave-valor, con 100 pares como máximo.

  Cada clave es una cadena UTF-8.

  Cada valor es una cadena UTF-8.

  Mapa de argumentos que se utiliza para configurar `DevEndpoint`.

Respuesta

- `EndpointName`: cadena UTF-8.

  El nombre asignado a la nueva `DevEndpoint`.

- `Status`: cadena UTF-8.

El estado actual del nuevo `DevEndpoint`.

- `SecurityGroupIds`: matriz de cadenas UTF-8.

  Los grupos de seguridad que se asignaron al nuevo `DevEndpoint`.

- `SubnetId`: cadena UTF-8.

  El ID de subred asignado al nuevo `DevEndpoint`.

- `RoleArn`: cadena UTF-8 que coincide con el AWS IAM ARN string pattern.

  El nombre de recurso de Amazon (ARN) del rol asignado al nuevo `DevEndpoint`.

- `YarnEndpointAddress`: cadena UTF-8.

  La dirección del punto de enlace de YARN que utiliza este `DevEndpoint`.

- `ZeppelinRemoteSparkInterpreterPort`: número (entero).

  El puerto de Apache Zeppelin para el intérprete remoto de Apache Spark.

- `NumberOfNodes`: número (entero).

  El número de unidades de procesamiento de datos de AWS Glue (DPU) asignadas a este punto de enlace de desarrollo.

- `WorkerType`: cadena UTF-8 (valores válidos: `Standard=""` | `G.1X=""` | `G.2X=""` | `G.025X=""` | `G.4X=""` | `G.8X=""` | `Z.2X=""`).

  El tipo de proceso de trabajo predefinido que se asigna al punto de enlace de desarrollo. Puede ser un valor de Standard, G.1X o G.2X.

- `GlueVersion`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Custom string pattern #20.

  La versión de Glue determina las versiones de Apache Spark y Python que admite AWS Glue. La versión de Python indica la versión admitida para ejecutar sus scripts de ETL en puntos de enlace de desarrollo.

  Para obtener más información acerca de las versiones de AWS Glue disponibles y las versiones de Spark y Python correspondientes, consulte Versión de Glue en la guía para desarrolladores.

- `NumberOfWorkers`: número (entero).

El número de procesos de trabajo de un definido `workerType` que se asignan al punto de enlace de desarrollo.

- `AvailabilityZone`: cadena UTF-8.

  La zona de disponibilidad de AWS en la que se encuentra este `DevEndpoint`.

- `VpcId`: cadena UTF-8.

  El ID de la nube privada virtual (VPC) que utiliza este `DevEndpoint`.

- `ExtraPythonLibsS3Path`: cadena UTF-8.

  Las rutas a una o varias bibliotecas Python en un bucket de S3 que se cargarán en el `DevEndpoint`.

- `ExtraJarsS3Path`: cadena UTF-8.

  Ruta a uno o varios archivos `.jar` de Java en un bucket de S3 que se cargarán en el `DevEndpoint`.

- `FailureReason`: cadena UTF-8.

  El motivo de un error actual en este `DevEndpoint`.

- `SecurityConfiguration`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la estructura `SecurityConfiguration` que se va a utilizar con este `DevEndpoint`.

- `CreatedTimestamp`: marca temporal.

  El punto en el que se creó este `DevEndpoint`.

- `Arguments`: matriz de mapas de pares de clave-valor, con 100 pares como máximo.

  Cada clave es una cadena UTF-8.

  Cada valor es una cadena UTF-8.

  Mapa de argumentos que se utiliza para configurar este `DevEndpoint`.

  Los argumentos válidos son:

  - `"--enable-glue-datacatalog": ""`

Puede especificar una versión de Python compatible con puntos de enlace de desarrollo mediante el parámetro `Arguments` en las API `UpdateDevEndpoint` o `CreateDevEndpoint`. Si no se proporcionan argumentos, la versión es Python 2 de forma predeterminada.

Errores

- `AccessDeniedException`
- `AlreadyExistsException`
- `IdempotentParameterMismatchException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`
- `ValidationException`
- `ResourceNumberLimitExceededException`

Acción UpdateDevEndpoint (Python: update_dev_endpoint)

Actualiza un punto de enlace de desarrollo especificado.

Solicitud

- `EndpointName` – Obligatorio: cadena UTF-8.

  El nombre del `DevEndpoint` que actualizar.
- `PublicKey`: cadena UTF-8.

  La clave pública del `DevEndpoint` que utilizar.
- `AddPublicKeys`: matriz de cadenas UTF-8, con 5 cadenas como máximo.

  La lista de las claves públicas del `DevEndpoint` que se va a utilizar.
- `DeletePublicKeys`: matriz de cadenas UTF-8, con 5 cadenas como máximo.

  Lista de claves públicas que se van a eliminar del `DevEndpoint`.
- `CustomLibraries`: un objeto DevEndpointCustomLibraries.

Las bibliotecas Python o Java personalizadas que se van a cargar en el `DevEndpoint`.

- `UpdateEtlLibraries`: booleano.

  `True` si la lista de bibliotecas personalizadas que se va a cargar el punto de enlace de desarrollo debe actualizarse; `False` en caso contrario.

- `DeleteArguments`: matriz de cadenas UTF-8.

  Lista de claves de argumentos que se van a eliminar del mapa de argumentos utilizado para configurar el objeto `DevEndpoint`.

- `AddArguments`: matriz de mapas de pares de clave-valor, con 100 pares como máximo.

  Cada clave es una cadena UTF-8.

  Cada valor es una cadena UTF-8.

  Mapa de argumentos que se va a añadir al mapa de argumentos utilizado para configurar el `DevEndpoint`.

  Los argumentos válidos son:

  - `"--enable-glue-datacatalog": ""`

  Puede especificar una versión de Python compatible con puntos de enlace de desarrollo mediante el parámetro `Arguments` en las API `UpdateDevEndpoint` o `CreateDevEndpoint`. Si no se proporcionan argumentos, la versión es Python 2 de forma predeterminada.

Respuesta

- Sin parámetros de respuesta.

Errores

- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`
- `ValidationException`

Acción DeleteDevEndpoint (Python: delete_dev_endpoint)

Elimina un punto de enlace de desarrollo especificado.

Solicitud

- `EndpointName` – Obligatorio: cadena UTF-8.

  Nombre del elemento `DevEndpoint`.

Respuesta

- Sin parámetros de respuesta.

Errores

- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`

Acción GetDevEndpoint (Python: get_dev_endpoint)

Recupera información sobre un punto de enlace de desarrollo especificado.

> **Note**
>
> Cuando se crea un punto de enlace de desarrollo en una nube virtual privada (VPC), AWS Glue devuelve solamente una dirección IP privada y el campo de la dirección IP pública no se rellena. Al crear un punto de enlace de desarrollo que no pertenece a la VPC, AWS Glue devuelve solamente una dirección IP pública.

Solicitud

- `EndpointName` – Obligatorio: cadena UTF-8.

  Nombre del `DevEndpoint` para el que recuperar información.

Respuesta

- `DevEndpoint`: un objeto DevEndpoint.

  Una definición de `DevEndpoint`.

Errores

- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`

Acción GetDevEndpoints (Python: get_dev_endpoints)

Recupera todos los puntos de enlace de desarrollo en esta cuenta de AWS.

> **Note**
>
> Cuando se crea un punto de enlace de desarrollo en una nube virtual privada (VPC), AWS Glue devuelve solamente una dirección IP privada y el campo de la dirección IP pública no se rellena. Al crear un punto de enlace de desarrollo que no pertenece a la VPC, AWS Glue devuelve solamente una dirección IP pública.

Solicitud

- `MaxResults`: número (entero) que no es inferior a 1 ni es superior a 1000.

  Tamaño máximo de la información que se devolverá.
- `NextToken`: cadena UTF-8.

  Token de continuación si se trata de una llamada de continuidad.

Respuesta

- `DevEndpoints`: matriz de objetos DevEndpoint.

Una lista de definiciones de `DevEndpoint`.

- `NextToken`: cadena UTF-8.

  Un token de continuación si todavía no se han devuelto todas las definiciones de `DevEndpoint`.

Errores

- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`

Acción BatchGetDevEndpoints (Python: batch_get_dev_endpoints)

Devuelve la lista de metadatos de recursos de una determinada lista de nombres de punto de enlace de desarrollo. Después de llamar a la operación `ListDevEndpoints`, puede llamar a esta operación para obtener acceso a los datos a los que ha concedido permisos. Esta operación admite todos los permisos de IAM, incluidas las condiciones de permisos que utilizan etiquetas.

Solicitud

- `customerAccountId`: cadena UTF-8.

  El ID de la cuenta de AWS.

- `DevEndpointNames` – Obligatorio: matriz de cadenas UTF-8, con una cadena como mínimo y 25 cadenas como máximo.

  La lista de nombres de `DevEndpoint`, que podrían ser los nombres devueltos desde la operación `ListDevEndpoint`.

Respuesta

- `DevEndpoints`: matriz de objetos DevEndpoint.

  Una lista de definiciones de `DevEndpoint`.

- `DevEndpointsNotFound`: matriz de cadenas UTF-8, con una cadena como mínimo y 25 cadenas como máximo.

  Una lista de `DevEndpoints` no encontrados.

Errores

- `AccessDeniedException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`

Acción ListDevEndpoints (Python: list_dev_endpoints)

Recupera los nombres de todos los recursos de `DevEndpoint` en esta cuenta de AWS, o los recursos con la etiqueta especificada. Esta operación permite ver qué recursos están disponibles en la cuenta y sus nombres.

Esta operación toma el campo `Tags` opcional, que se puede utilizar como filtro en la respuesta para que los recursos etiquetados se devuelvan agrupados. Si decide utilizar el filtrado de etiquetas, solo se devolverán los recursos con la etiqueta especificada.

Solicitud

- `NextToken`: cadena UTF-8.

  Token de continuación, si se trata de una solicitud de continuidad.
- `MaxResults`: número (entero) que no es inferior a 1 ni es superior a 1000.

  Tamaño máximo de una lista que se devolverá.
- `Tags`: matriz de mapas de pares de clave-valor, con 50 pares como máximo.

  Cada clave es una cadena UTF-8 con una longitud de entre 1 y 128 bytes.

  Cada valor es una cadena UTF-8 que no tiene más de 256 bytes de largo.

  Especifica que se devuelvan solamente los recursos etiquetados.

Respuesta

- `DevEndpointNames`: matriz de cadenas UTF-8.

  Nombres de todos los `DevEndpoint` de la cuenta o de los `DevEndpoint` con las etiquetas especificadas.
- `NextToken`: cadena UTF-8.

  Token de continuación, si la lista devuelta no contiene la última métrica disponible.

Errores

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`

Schema Registry

La API de registro de esquemas describe los tipos de datos y la API relacionados con el trabajo con esquemas en AWS Glue.

Tipos de datos

- RegistryId estructura
- RegistryListItem estructura
- MetadataInfo estructura
- OtherMetadataValueListItem estructura
- SchemaListItem estructura
- SchemaVersionListItem estructura
- MetadataKeyValuePair estructura
- SchemaVersionErrorItem estructura
- ErrorDetails estructura
- SchemaVersionNumber estructura
- SchemaId estructura

RegistryId estructura

Una estructura de encapsulador que puede contener el nombre de registro y el nombre de recurso de Amazon (ARN).

Campos

- `RegistryName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Custom string pattern #18.

  Nombre del registro. Sólo se utiliza para la búsqueda. Uno de `RegistryArn` o `RegistryName` debe ser proporcionado.
- `RegistryArn`: cadena UTF-8, con 1 byte de largo como mínimo y 10 240 bytes de largo como máximo, que coincide con el Custom string pattern #22.

  Arn del registro que se actualizará. Uno de `RegistryArn` o `RegistryName` debe ser proporcionado.

RegistryListItem estructura

Una estructura que contiene los detalles de un registro.

Campos

- `RegistryName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Custom string pattern #18.

  El nombre del registro.
- `RegistryArn`: cadena UTF-8, con 1 byte de largo como mínimo y 10 240 bytes de largo como máximo, que coincide con el Custom string pattern #22.

  Nombre de recurso de Amazon (ARN) del registro.
- `Description`: cadena de descripción de un máximo de 2048 bytes de largo, que coincide con el URI address multi-line string pattern.

  Es una descripción del recurso.
- `Status`: cadena UTF-8 (valores válidos: `AVAILABLE` | `DELETING`).

  Estado del registro.

- `CreatedTime`: cadena UTF-8.

  La fecha en que se creó el registro.
- `UpdatedTime`: cadena UTF-8.

  La fecha en que se actualizó el registro.

MetadataInfo estructura

Estructura que contiene información de metadatos para una versión de esquema.

Campos

- `MetadataValue`: cadena UTF-8, con 1 byte de largo como mínimo y 256 bytes de largo como máximo, que coincide con el Custom string pattern #33.

  Valor correspondiente de una clave de metadatos.
- `CreatedTime`: cadena UTF-8.

  La hora a la que se creó la entrada.
- `OtherMetadataValueList`: matriz de objetos OtherMetadataValueListItem.

  Otros metadatos pertenecientes a la misma clave de metadatos.

OtherMetadataValueListItem estructura

Estructura que contiene otros metadatos para una versión de esquema que pertenece a la misma clave de metadatos.

Campos

- `MetadataValue`: cadena UTF-8, con 1 byte de largo como mínimo y 256 bytes de largo como máximo, que coincide con el Custom string pattern #33.

  El valor correspondiente de la clave de metadatos para los demás metadatos pertenecientes a la misma clave de metadatos.
- `CreatedTime`: cadena UTF-8.

  La hora a la que se creó la entrada.

SchemaListItem estructura

Objeto que contiene detalles mínimos de un esquema.

Campos

- `RegistryName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Custom string pattern #18.

  Nombre del registro donde reside el esquema.

- `SchemaName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Custom string pattern #18.

  El nombre del esquema.

- `SchemaArn`: cadena UTF-8, con 1 byte de largo como mínimo y 10 240 bytes de largo como máximo, que coincide con el Custom string pattern #22.

  Nombre de recurso de Amazon (ARN) del esquema.

- `Description`: cadena de descripción de un máximo de 2048 bytes de largo, que coincide con el URI address multi-line string pattern.

  Una descripción del esquema.

- `SchemaStatus`: cadena UTF-8 (valores válidos: `AVAILABLE` | `PENDING` | `DELETING`).

  Estado del esquema.

- `CreatedTime`: cadena UTF-8.

  La fecha y hora en que se creó el esquema.

- `UpdatedTime`: cadena UTF-8.

  La fecha y hora en que se actualizó el esquema.

SchemaVersionListItem estructura

Objeto que contiene los detalles sobre una versión de esquema.

Campos

- `SchemaArn`: cadena UTF-8, con 1 byte de largo como mínimo y 10 240 bytes de largo como máximo, que coincide con el Custom string pattern #22.

  Nombre de recurso de Amazon (ARN) del esquema.
- `SchemaVersionId`: cadena UTF-8, con 36 bytes de largo como mínimo y 36 bytes de largo como máximo, que coincide con el Custom string pattern #17.

  El identificador único de la versión del esquema.
- `VersionNumber`: número (largo) que no es inferior a 1 ni es superior a 100 000.

  El número de versión del esquema.
- `Status`: cadena UTF-8 (valores válidos: `AVAILABLE` | `PENDING` | `FAILURE` | `DELETING`).

  El estado de la versión del esquema.
- `CreatedTime`: cadena UTF-8.

  La fecha y hora en que se creó la versión del esquema.

MetadataKeyValuePair estructura

Una estructura que contiene un par de valor-clave para metadatos.

Campos

- `MetadataKey`: cadena UTF-8, con 1 byte de largo como mínimo y 128 bytes de largo como máximo, que coincide con el Custom string pattern #33.

  Una clave de metadatos.
- `MetadataValue`: cadena UTF-8, con 1 byte de largo como mínimo y 256 bytes de largo como máximo, que coincide con el Custom string pattern #33.

  Valor correspondiente de una clave de metadatos.

SchemaVersionErrorItem estructura

Objeto que contiene los detalles de error de una operación en una versión de esquema.

Campos

- `VersionNumber`: número (largo) que no es inferior a 1 ni es superior a 100 000.

  El número de versión del esquema.
- `ErrorDetails`: un objeto ErrorDetails.

  Detalles del error para la versión del esquema.

ErrorDetails estructura

Objeto que contiene detalles de error.

Campos

- `ErrorCode`: cadena UTF-8.

  El código de error de un error.
- `ErrorMessage`: cadena UTF-8.

  El mensaje de un error.

SchemaVersionNumber estructura

Una estructura que contiene la información de versión del esquema.

Campos

- `LatestVersion`: booleano.

  La última versión disponible para el esquema.
- `VersionNumber`: número (largo) que no es inferior a 1 ni es superior a 100 000.

  El número de versión del esquema.

SchemaId estructura

El identificador único del esquema en el registro AWS Glue de esquemas.

Campos

- `SchemaArn`: cadena UTF-8, con 1 byte de largo como mínimo y 10 240 bytes de largo como máximo, que coincide con el Custom string pattern #22.

  Nombre de recurso de Amazon (ARN) del esquema. Uno de `SchemaArn` o `SchemaName` debe ser proporcionado.
- `SchemaName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Custom string pattern #18.

  El nombre del esquema. Uno de `SchemaArn` o `SchemaName` debe ser proporcionado.
- `RegistryName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Custom string pattern #18.

  Nombre del registro de esquemas que contiene el esquema.

Operaciones

- CreateRegistry acción (Python: create_registry)
- CreateSchema acción (Python: create_schema)
- GetSchema acción (Python: get_schema)
- ListSchemaVersions acción (Python: list_schema_versions)
- GetSchemaVersion acción (Python: get_schema_version)
- GetSchemaVersionsDiff acción (Python: get_schema_versions_diff)
- ListRegistries acción (Python: list_registries)
- ListSchemas acción (Python: list_schemas)
- RegisterSchemaVersion acción (Python: register_schema_version)
- UpdateSchema acción (Python: update_schema)
- CheckSchemaVersionValidity acción (Python: check_schema_version_valid)
- UpdateRegistry acción (Python: update_registry)
- GetSchemaByDefinition acción (Python: get_schema_by_definition)
- GetRegistry acción (Python: get_registry)
- PutSchemaVersionMetadata acción (Python: put_schema_version_metadata)
- QuerySchemaVersionMetadata acción (Python: query_schema_version_metadata)

- RemoveSchemaVersionMetadata acción (Python: remove_schema_version_metadata)
- DeleteRegistry acción (Python: delete_registry)
- DeleteSchema acción (Python: delete_schema)
- DeleteSchemaVersions acción (Python: delete_schema_versions)

CreateRegistry acción (Python: create_registry)

Crea un nuevo registro que se puede utilizar para contener una recopilación de esquemas.

Solicitud

- `RegistryName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Custom string pattern #18.

  Nombre del registro que se va a crear con una longitud máxima de 255, solo puede contener letras, números, guión, guión bajo, signo de dólar o marca hash. Sin espacio en blanco.
- `Description`: cadena de descripción de un máximo de 2048 bytes de largo, que coincide con el URI address multi-line string pattern.

  Es una descripción del recurso. Si no se proporciona una descripción, no habrá ningún valor predeterminado para esto.
- `Tags`: matriz de mapas de pares de clave-valor, con 50 pares como máximo.

  Cada clave es una cadena UTF-8 con una longitud de entre 1 y 128 bytes.

  Cada valor es una cadena UTF-8 que no tiene más de 256 bytes de largo.

  AWS etiquetas que contienen un par clave-valor y que se pueden buscar por consola, línea de comandos o API.

Respuesta

- `RegistryArn`: cadena UTF-8, con 1 byte de largo como mínimo y 10 240 bytes de largo como máximo, que coincide con el Custom string pattern #22.

  El nombre de recurso de Amazon (ARN) del registro recientemente creado.
- `RegistryName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Custom string pattern #18.

El nombre del registro.

- `Description`: cadena de descripción de un máximo de 2048 bytes de largo, que coincide con el URI address multi-line string pattern.

  Es una descripción del recurso.

- `Tags`: matriz de mapas de pares de clave-valor, con 50 pares como máximo.

  Cada clave es una cadena UTF-8 con una longitud de entre 1 y 128 bytes.

  Cada valor es una cadena UTF-8 que no tiene más de 256 bytes de largo.

  Etiquetas para el registro.

Errores

- `InvalidInputException`
- `AccessDeniedException`
- `AlreadyExistsException`
- `ResourceNumberLimitExceededException`
- `ConcurrentModificationException`
- `InternalServiceException`

CreateSchema acción (Python: create_schema)

Crea un nuevo conjunto de esquemas y registra la definición de esquema. Devuelve un error si el conjunto de esquemas ya existe sin haber registrado realmente la versión.

Cuando se crea el conjunto de esquemas, se establecerá un punto de comprobación de versión en la primera versión. El modo de compatibilidad “DISABLED (DESACTIVADO)” restringe el agregado de versiones de esquema adicionales después de la primera versión del esquema. Para todos los demás modos de compatibilidad, la validación de la configuración de compatibilidad solo se aplicará a partir de la segunda versión cuando se utilice la API de `RegisterSchemaVersion`.

Cuando se llama a esta API sin un `RegistryId`, esto creará una entrada para un “registro predeterminado” en las tablas de la base de datos del registro, si aún no se encuentra incluido.

Solicitud

- `RegistryId`: un objeto RegistryId.

  Esta es una forma de encapsulador que contiene los campos de identidad del registro. Si no se proporciona, se usará el registro predeterminado. El formato del ARN para el mismo será: `arn:aws:glue:us-east-2:<customer id>:registry/default-registry:random-5-letter-id`.

- `SchemaName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Custom string pattern #18.

  Nombre del esquema que se va a crear con una longitud máxima de 255, y solo puede contener letras, números, guión, guión bajo, signo de dólar o marca hash. Sin espacio en blanco.

- `DataFormat` – Obligatorio: cadena UTF-8 (valores válidos: `AVRO` | `JSON` | `PROTOBUF`).

  El formato de datos de la definición de esquema. `AVRO`, `JSON` y `PROTOBUF` se admiten en la actualidad.

- `Compatibility`: cadena UTF-8 (valores válidos: `NONE` | `DISABLED` | `BACKWARD` | `BACKWARD_ALL` | `FORWARD` | `FORWARD_ALL` | `FULL` | `FULL_ALL`).

  El modo de compatibilidad del esquema. Los valores posibles son:

  - NONE (NINGUNO): no se aplica ningún modo de compatibilidad. Puede utilizar esta opción en escenarios de desarrollo o si no conoce el modo de compatibilidad que desea aplicar a los esquemas. Cualquier nueva versión que se agregue será aceptada sin someterse a una comprobación de compatibilidad.
  - DISABLED (DESACTIVADO): esta opción de compatibilidad impide el control de versiones de un esquema en particular. Puede utilizar esta opción para evitar el control de versiones de un esquema a futuro.
  - BACKWARDS (HACIA ATRÁS): se recomienda esta opción de compatibilidad ya que permite a los receptores de datos leer tanto la versión actual como la anterior del esquema. Esto significa que, por ejemplo, una nueva versión de esquema no puede eliminar campos de datos ni cambiar el tipo de estos campos, por lo que los lectores no pueden leerlos con la versión anterior.
  - BACKWARD_ALL (HACIA ATRÁS_TODO): esta opción de compatibilidad permite a los receptores de datos leer tanto la versión actual como todas las versiones anteriores del esquema. Puede utilizar esta opción cuando necesite eliminar campos o agregar campos opcionales, y comprobar la compatibilidad con todas las versiones de esquema anteriores.

  - FORWARD (HACIA ADELANTE): esta opción de compatibilidad permite a los receptores de datos leer tanto la versión actual como la inmediatamente siguiente del esquema, pero no necesariamente versiones posteriores. Puede utilizar esta opción cuando necesite agregar campos o eliminar campos opcionales, pero comprobar la compatibilidad con la última versión del esquema únicamente.
  - FORWARD_ALL (HACIA ADELANTE_TODO): esta opción de compatibilidad permite a los receptores de datos leer datos escritos por los productores de cualquier nuevo esquema registrado. Puede utilizar esta opción cuando necesite agregar campos o eliminar campos opcionales, y comprobar la compatibilidad con todas las versiones de esquema anteriores.
  - FULL (COMPLETO): esta opción de compatibilidad permite a los receptores de datos leer los datos escritos por los productores utilizando la versión inmediatamente anterior o siguiente del esquema, pero no necesariamente versiones anteriores o posteriores. Puede utilizar esta opción cuando necesite agregar campos o eliminar campos opcionales, pero comprobar la compatibilidad con la última versión del esquema únicamente.
  - FULL_ALL (COMPLETO_TODO): esta opción de compatibilidad permite a los receptores de datos leer los datos escritos por los productores utilizando todas las versiones de esquema anteriores. Puede utilizar esta opción cuando necesite agregar campos o eliminar campos opcionales, y comprobar la compatibilidad con todas las versiones anteriores del esquema.
- `Description`: cadena de descripción de un máximo de 2048 bytes de largo, que coincide con el URI address multi-line string pattern.

  Una descripción opcional del esquema. Si no se proporciona una descripción, no habrá ningún valor predeterminado automático para esto.
- `Tags`: matriz de mapas de pares de clave-valor, con 50 pares como máximo.

  Cada clave es una cadena UTF-8 con una longitud de entre 1 y 128 bytes.

  Cada valor es una cadena UTF-8 que no tiene más de 256 bytes de largo.

  AWS etiquetas que contienen un par clave-valor y que se pueden buscar por consola, línea de comandos o API. Si se especifica, sigue el AWS tags-on-create patrón.
- `SchemaDefinition`: cadena UTF-8, con 1 byte de largo como mínimo y 170 000 bytes de largo como máximo, que coincide con el Custom string pattern #32.

  Definición de esquema que utiliza la configuración `DataFormat` de `SchemaName`.

Respuesta

- `RegistryName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Custom string pattern #18.

  El nombre del registro.
- `RegistryArn`: cadena UTF-8, con 1 byte de largo como mínimo y 10 240 bytes de largo como máximo, que coincide con el Custom string pattern #22.

  Nombre de recurso de Amazon (ARN) del registro.
- `SchemaName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Custom string pattern #18.

  El nombre del esquema.
- `SchemaArn`: cadena UTF-8, con 1 byte de largo como mínimo y 10 240 bytes de largo como máximo, que coincide con el Custom string pattern #22.

  Nombre de recurso de Amazon (ARN) del esquema.
- `Description`: cadena de descripción de un máximo de 2048 bytes de largo, que coincide con el URI address multi-line string pattern.

  Una descripción del esquema si se especifica cuando se crea.
- `DataFormat`: cadena UTF-8 (valores válidos: `AVRO` | `JSON` | `PROTOBUF`).

  El formato de datos de la definición de esquema. `AVRO`, `JSON` y `PROTOBUF` se admiten en la actualidad.
- `Compatibility`: cadena UTF-8 (valores válidos: `NONE` | `DISABLED` | `BACKWARD` | `BACKWARD_ALL` | `FORWARD` | `FORWARD_ALL` | `FULL` | `FULL_ALL`).

  El modo de compatibilidad del esquema.
- `SchemaCheckpoint`: número (largo) que no es inferior a 1 ni es superior a 100 000.

  El número de versión del punto de comprobación (la última vez que se cambió el modo de compatibilidad).
- `LatestSchemaVersion`: número (largo) que no es inferior a 1 ni es superior a 100 000.

  La última versión del esquema asociada a la definición del esquema devuelta.
- `NextSchemaVersion`: número (largo) que no es inferior a 1 ni es superior a 100 000.

La siguiente versión del esquema asociada a la definición de esquema devuelta.

- `SchemaStatus`: cadena UTF-8 (valores válidos: `AVAILABLE` | `PENDING` | `DELETING`).

  Estado del esquema.

- `Tags`: matriz de mapas de pares de clave-valor, con 50 pares como máximo.

  Cada clave es una cadena UTF-8 con una longitud de entre 1 y 128 bytes.

  Cada valor es una cadena UTF-8 que no tiene más de 256 bytes de largo.

  Etiquetas para el esquema.

- `SchemaVersionId`: cadena UTF-8, con 36 bytes de largo como mínimo y 36 bytes de largo como máximo, que coincide con el Custom string pattern #17.

  El identificador único de la primera versión del esquema.

- `SchemaVersionStatus`: cadena UTF-8 (valores válidos: `AVAILABLE` | `PENDING` | `FAILURE` | `DELETING`).

  El estado de la primera versión creada del esquema.

Errores

- `InvalidInputException`
- `AccessDeniedException`
- `EntityNotFoundException`
- `AlreadyExistsException`
- `ResourceNumberLimitExceededException`
- `ConcurrentModificationException`
- `InternalServiceException`

GetSchema acción (Python: get_schema)

Describe el esquema especificado en detalle.

Solicitud

- `SchemaId`: obligatorio: objeto SchemaId.

  Una estructura de encapsulador que contiene campos de identidad de esquema. La estructura contiene lo siguiente:

  - SchemaId$SchemaArn: el nombre de recurso de Amazon (ARN) del esquema. Se debe proporcionar `SchemaArn` o `SchemaName` y `RegistryName`.
  - SchemaId$SchemaName: el nombre del esquema. Se debe proporcionar `SchemaArn` o `SchemaName` y `RegistryName`.

Respuesta

- `RegistryName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Custom string pattern #18.

  El nombre del registro.
- `RegistryArn`: cadena UTF-8, con 1 byte de largo como mínimo y 10 240 bytes de largo como máximo, que coincide con el Custom string pattern #22.

  Nombre de recurso de Amazon (ARN) del registro.
- `SchemaName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Custom string pattern #18.

  El nombre del esquema.
- `SchemaArn`: cadena UTF-8, con 1 byte de largo como mínimo y 10 240 bytes de largo como máximo, que coincide con el Custom string pattern #22.

  Nombre de recurso de Amazon (ARN) del esquema.
- `Description`: cadena de descripción de un máximo de 2048 bytes de largo, que coincide con el URI address multi-line string pattern.

  Se especifica una descripción del esquema en el momento de la creación
- `DataFormat`: cadena UTF-8 (valores válidos: `AVRO` | `JSON` | `PROTOBUF`).

  El formato de datos de la definición de esquema. `AVRO`, `JSON` y `PROTOBUF` se admiten en la actualidad.

- `Compatibility`: cadena UTF-8 (valores válidos: `NONE` | `DISABLED` | `BACKWARD` | `BACKWARD_ALL` | `FORWARD` | `FORWARD_ALL` | `FULL` | `FULL_ALL`).

  El modo de compatibilidad del esquema.

- `SchemaCheckpoint`: número (largo) que no es inferior a 1 ni es superior a 100 000.

  El número de versión del punto de comprobación (la última vez que se cambió el modo de compatibilidad).

- `LatestSchemaVersion`: número (largo) que no es inferior a 1 ni es superior a 100 000.

  La última versión del esquema asociada a la definición del esquema devuelta.

- `NextSchemaVersion`: número (largo) que no es inferior a 1 ni es superior a 100 000.

  La siguiente versión del esquema asociada a la definición de esquema devuelta.

- `SchemaStatus`: cadena UTF-8 (valores válidos: `AVAILABLE` | `PENDING` | `DELETING`).

  Estado del esquema.

- `CreatedTime`: cadena UTF-8.

  La fecha y hora en que se creó el esquema.

- `UpdatedTime`: cadena UTF-8.

  La fecha y hora en que se actualizó el esquema.

Errores

- `InvalidInputException`
- `AccessDeniedException`
- `EntityNotFoundException`
- `InternalServiceException`

ListSchemaVersions acción (Python: list_schema_versions)

Devuelve una lista de versiones de esquema que ha creado, con información mínima. Las versiones del esquema en estado Deleted (Eliminado) no se incluirán en los resultados. Los resultados vacíos se devolverán si no hay versiones de esquema disponibles.

Solicitud

- `SchemaId`: obligatorio: objeto SchemaId.

  Una estructura de encapsulador que contiene campos de identidad de esquema. La estructura contiene lo siguiente:

  - SchemaId$SchemaArn: el nombre de recurso de Amazon (ARN) del esquema. Se debe proporcionar `SchemaArn` o `SchemaName` y `RegistryName`.
  - SchemaId$SchemaName: el nombre del esquema. Se debe proporcionar `SchemaArn` o `SchemaName` y `RegistryName`.
- `MaxResults`: número (entero) que no es inferior a 1 ni es superior a 100.

  Número máximo de resultados requeridos por página. Si no se proporciona el valor, el valor predeterminado será 25 por página.
- `NextToken`: cadena UTF-8.

  Token de continuación si se trata de una llamada de continuidad.

Respuesta

- `Schemas`: matriz de objetos SchemaVersionListItem.

  Una matriz de objetos de `SchemaVersionList` que contiene detalles de cada versión del esquema.
- `NextToken`: cadena UTF-8.

  Token de continuación para paginar la lista de tokens obtenida; se devuelve si el segmento actual de la lista no es el último.

Errores

- `InvalidInputException`
- `AccessDeniedException`
- `EntityNotFoundException`
- `InternalServiceException`

GetSchemaVersion acción (Python: get_schema_version)

Obtenga el esquema especificado por su ID único asignado cuando se crea o registra una versión del esquema. Las versiones del esquema en estado Deleted (Eliminado) no se incluirán en los resultados.

Solicitud

- `SchemaId`: un objeto SchemaId.

  Una estructura de encapsulador que contiene campos de identidad de esquema. La estructura contiene lo siguiente:

  - SchemaId$SchemaArn: el nombre de recurso de Amazon (ARN) del esquema. Se debe proporcionar `SchemaArn` o `SchemaName` y `RegistryName`.
  - SchemaId$SchemaName: el nombre del esquema. Se debe proporcionar `SchemaArn` o `SchemaName` y `RegistryName`.
- `SchemaVersionId`: cadena UTF-8, con 36 bytes de largo como mínimo y 36 bytes de largo como máximo, que coincide con el Custom string pattern #17.

  El `SchemaVersionId` de versión del esquema. Este campo es obligatorio para recuperar por ID de esquema. Se debe proporcionar este encapsulador o `SchemaId`.
- `SchemaVersionNumber`: un objeto SchemaVersionNumber.

  El número de versión del esquema.

Respuesta

- `SchemaVersionId`: cadena UTF-8, con 36 bytes de largo como mínimo y 36 bytes de largo como máximo, que coincide con el Custom string pattern #17.

  El `SchemaVersionId` de versión del esquema.
- `SchemaDefinition`: cadena UTF-8, con 1 byte de largo como mínimo y 170 000 bytes de largo como máximo, que coincide con el Custom string pattern #32.

  Definición de esquema para ID del esquema.
- `DataFormat`: cadena UTF-8 (valores válidos: `AVRO` | `JSON` | `PROTOBUF`).

El formato de datos de la definición de esquema. `AVRO`, `JSON` y `PROTOBUF` se admiten en la actualidad.

- `SchemaArn`: cadena UTF-8, con 1 byte de largo como mínimo y 10 240 bytes de largo como máximo, que coincide con el Custom string pattern #22.

  Nombre de recurso de Amazon (ARN) del esquema.
- `VersionNumber`: número (largo) que no es inferior a 1 ni es superior a 100 000.

  El número de versión del esquema.
- `Status`: cadena UTF-8 (valores válidos: `AVAILABLE` | `PENDING` | `FAILURE` | `DELETING`).

  El estado de la versión del esquema.
- `CreatedTime`: cadena UTF-8.

  La fecha y hora en que se creó la versión del esquema.

Errores

- `InvalidInputException`
- `AccessDeniedException`
- `EntityNotFoundException`
- `InternalServiceException`

GetSchemaVersionsDiff acción (Python: get_schema_versions_diff)

Recupera la diferencia de versión del esquema en el tipo de diferencia especificado entre dos versiones de esquema almacenadas en el Schema Registry.

Esta API le permite comparar dos versiones entre dos definiciones de esquemas bajo el mismo esquema.

Solicitud

- `SchemaId`: obligatorio: objeto SchemaId.

  Una estructura de encapsulador que contiene campos de identidad de esquema. La estructura contiene lo siguiente:

  - SchemaId$SchemaArn: el nombre de recurso de Amazon (ARN) del esquema. Uno de `SchemaArn` o `SchemaName` debe ser proporcionado.
  - SchemaId$SchemaName: el nombre del esquema. Uno de `SchemaArn` o `SchemaName` debe ser proporcionado.
- `FirstSchemaVersionNumber`: obligatorio: objeto SchemaVersionNumber.

  La primera de las dos versiones de esquema que se van a comparar.
- `SecondSchemaVersionNumber`: obligatorio: objeto SchemaVersionNumber.

  La segunda de las dos versiones de esquema que se van a comparar.
- `SchemaDiffType` – Obligatorio: cadena UTF-8 (valores válidos: `SYNTAX_DIFF`).

  Se refiere a `SYNTAX_DIFF`, que es el tipo diferente que se soporta actualmente.

Respuesta

- `Diff`: cadena UTF-8, con 1 byte de largo como mínimo y 340 000 bytes de largo como máximo, que coincide con el Custom string pattern #32.

  La diferencia entre los esquemas en JsonPatch formato de cadena.

Errores

- `InvalidInputException`
- `EntityNotFoundException`
- `AccessDeniedException`
- `InternalServiceException`

ListRegistries acción (Python: list_registries)

Devuelve una lista de registros que ha creado, con información mínima del registro. Los registros con estado `Deleting` no se incluirá en los resultados. Se devolverán resultados vacíos si no hay registros disponibles.

Solicitud

- `MaxResults`: número (entero) que no es inferior a 1 ni es superior a 100.

Número máximo de resultados requeridos por página. Si no se proporciona el valor, el valor predeterminado será 25 por página.

- `NextToken`: cadena UTF-8.

  Token de continuación si se trata de una llamada de continuidad.

Respuesta

- `Registries`: matriz de objetos RegistryListItem.

  Una matriz de objetos `RegistryDetailedListItem` que contienen detalles mínimos de cada registro.

- `NextToken`: cadena UTF-8.

  Token de continuación para paginar la lista de tokens obtenida; se devuelve si el segmento actual de la lista no es el último.

Errores

- `InvalidInputException`
- `AccessDeniedException`
- `InternalServiceException`

ListSchemas acción (Python: list_schemas)

Devuelve una lista de esquemas con detalles mínimos. Los esquemas con estado Deleting (Eliminación) no se incluirán en los resultados. Se devolverán resultados vacíos si no hay esquemas disponibles.

Cuando no se proporciona el `RegistryId`, todos los esquemas de los registros formarán parte de la respuesta de la API.

Solicitud

- `RegistryId`: un objeto RegistryId.

Una estructura de encapsulador que puede contener el nombre de registro y el nombre de recurso de Amazon (ARN).

- `MaxResults`: número (entero) que no es inferior a 1 ni es superior a 100.

  Número máximo de resultados requeridos por página. Si no se proporciona el valor, el valor predeterminado será 25 por página.

- `NextToken`: cadena UTF-8.

  Token de continuación si se trata de una llamada de continuidad.

Respuesta

- `Schemas`: matriz de objetos SchemaListItem.

  Una matriz de objetos `SchemaListItem` que contienen detalles de cada esquema.

- `NextToken`: cadena UTF-8.

  Token de continuación para paginar la lista de tokens obtenida; se devuelve si el segmento actual de la lista no es el último.

Errores

- `InvalidInputException`
- `AccessDeniedException`
- `EntityNotFoundException`
- `InternalServiceException`

RegisterSchemaVersion acción (Python: register_schema_version)

Agrega una nueva versión al esquema existente. Devuelve un error si la nueva versión del esquema no cumple los requisitos de compatibilidad del conjunto de esquemas. Esta API no creará un nuevo conjunto de esquemas y devolverá un error 404 si el conjunto de esquemas no está ya presente en el Schema Registry.

Si esta es la primera definición de esquema que se registra en el Schema Registry, esta API almacenará la versión del esquema y la devolverá inmediatamente. De lo contrario, esta llamada

tiene el potencial de ejecutarse más tiempo que otras operaciones debido a los modos de compatibilidad. Puede invocar la API de `GetSchemaVersion` con el `SchemaVersionId` para comprobar los modos de compatibilidad.

Si la misma definición de esquema ya está almacenada en el Schema Registry como una versión, el ID del esquema existente se devuelve al que realiza la llamada.

Solicitud

- `SchemaId`: obligatorio: objeto SchemaId.

  Una estructura de encapsulador que contiene campos de identidad de esquema. La estructura contiene lo siguiente:

  - SchemaId$SchemaArn: el nombre de recurso de Amazon (ARN) del esquema. Se debe proporcionar `SchemaArn` o `SchemaName` y `RegistryName`.
  - SchemaId$SchemaName: el nombre del esquema. Se debe proporcionar `SchemaArn` o `SchemaName` y `RegistryName`.
- `SchemaDefinition` – Obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 170 000 bytes de largo como máximo, que coincide con el Custom string pattern #32.

  Definición de esquema que utiliza la configuración `DataFormat` para el `SchemaName`.

Respuesta

- `SchemaVersionId`: cadena UTF-8, con 36 bytes de largo como mínimo y 36 bytes de largo como máximo, que coincide con el Custom string pattern #17.

  El ID único que representa la versión de este esquema.
- `VersionNumber`: número (largo) que no es inferior a 1 ni es superior a 100 000.

  La versión de este esquema (solo para flujo de sincronización, en caso de que sea la primera versión).
- `Status`: cadena UTF-8 (valores válidos: `AVAILABLE` | `PENDING` | `FAILURE` | `DELETING`).

  El estado de la versión del esquema.

Errores

- `InvalidInputException`
- `AccessDeniedException`
- `EntityNotFoundException`
- `ResourceNumberLimitExceededException`
- `ConcurrentModificationException`
- `InternalServiceException`

UpdateSchema acción (Python: update_schema)

Actualiza la descripción, la configuración de compatibilidad o el punto de comprobación de la versión de un conjunto de esquemas.

Para actualizar la configuración de compatibilidad, la llamada no validará la compatibilidad para todo el conjunto de versiones de esquema con la nueva configuración de compatibilidad. Si se proporciona el valor de `Compatibility`, también se requiere el `VersionNumber` (un punto de comprobación). La API validará el número de versión del punto de comprobación para fines de coherencia.

Si se proporciona el valor del `VersionNumber` (punto de comprobación), la `Compatibility` es opcional y esto se puede usar para establecer/restablecer un punto de comprobación para el esquema.

Esta actualización sólo se realizará si el esquema está en el estado AVAILABLE (DISPONIBLE).

Solicitud

- `SchemaId`: obligatorio: objeto SchemaId.

  Una estructura de encapsulador que contiene campos de identidad de esquema. La estructura contiene lo siguiente:

  - SchemaId$SchemaArn: el nombre de recurso de Amazon (ARN) del esquema. Uno de `SchemaArn` o `SchemaName` debe ser proporcionado.
  - SchemaId$SchemaName: el nombre del esquema. Uno de `SchemaArn` o `SchemaName` debe ser proporcionado.
- `SchemaVersionNumber`: un objeto SchemaVersionNumber.

Se requiere el número de versión para realizar la comprobación. Uno de `VersionNumber` o `Compatibility` debe ser proporcionado.

- `Compatibility`: cadena UTF-8 (valores válidos: `NONE` | `DISABLED` | `BACKWARD` | `BACKWARD_ALL` | `FORWARD` | `FORWARD_ALL` | `FULL` | `FULL_ALL`).

  La nueva configuración de compatibilidad para el esquema.

- `Description`: cadena de descripción de un máximo de 2048 bytes de largo, que coincide con el URI address multi-line string pattern.

  La nueva descripción del esquema.

Respuesta

- `SchemaArn`: cadena UTF-8, con 1 byte de largo como mínimo y 10 240 bytes de largo como máximo, que coincide con el Custom string pattern #22.

  Nombre de recurso de Amazon (ARN) del esquema.

- `SchemaName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Custom string pattern #18.

  El nombre del esquema.

- `RegistryName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Custom string pattern #18.

  Nombre del registro que contiene el esquema.

Errores

- `InvalidInputException`
- `AccessDeniedException`
- `EntityNotFoundException`
- `ConcurrentModificationException`
- `InternalServiceException`

CheckSchemaVersionValidity acción (Python: check_schema_version_valid)

Valida el esquema suministrado. Esta llamada no tiene efectos secundarios, simplemente valida el uso del esquema suministrado con `DataFormat` como el formato. Dado que no toma un nombre de conjunto de esquemas, no se realizan comprobaciones de compatibilidad.

Solicitud

- `DataFormat` – Obligatorio: cadena UTF-8 (valores válidos: `AVRO` | `JSON` | `PROTOBUF`).

  El formato de datos de la definición de esquema. `AVRO`, `JSON` y `PROTOBUF` se admiten en la actualidad.
- `SchemaDefinition` – Obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 170 000 bytes de largo como máximo, que coincide con el Custom string pattern #32.

  La definición del esquema que debe validarse.

Respuesta

- `Valid`: booleano.

  Devuelve verdadero, si el esquema es válido y falso en caso contrario.
- `Error`: cadena UTF-8, con 1 byte de largo como mínimo y 5000 bytes de largo como máximo.

  Mensaje de error por falla de validación.

Errores

- `InvalidInputException`
- `AccessDeniedException`
- `InternalServiceException`

UpdateRegistry acción (Python: update_registry)

Actualiza un registro existente que se utiliza para contener una recopilación de esquemas. Las propiedades actualizadas se relacionan con el registro y no modifican ninguno de los esquemas dentro del registro.

Solicitud

- `RegistryId`: obligatorio: objeto RegistryId.

  Una estructura de encapsulador que puede contener el nombre del registro y el nombre de recurso de Amazon (ARN).

- `Description` – Obligatorio: cadena de descripción, máximo de 2048 bytes de largo, que coincide con URI address multi-line string pattern.

  Es una descripción del recurso. Si no se proporciona una descripción, este campo no se actualizará.

Respuesta

- `RegistryName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Custom string pattern #18.

  El nombre del registro actualizado.

- `RegistryArn`: cadena UTF-8, con 1 byte de largo como mínimo y 10 240 bytes de largo como máximo, que coincide con el Custom string pattern #22.

  Nombre de recurso de Amazon (ARN) del registro actualizado.

Errores

- `InvalidInputException`
- `AccessDeniedException`
- `EntityNotFoundException`
- `ConcurrentModificationException`
- `InternalServiceException`

GetSchemaByDefinition acción (Python: get_schema_by_definition)

Recupera un esquema mediante la `SchemaDefinition`. La definición de esquema se envía a Schema Registry, se canonicaliza y se resume en un hash. Si el hash se hace coincidir dentro del alcance del `SchemaName` o ARN (o el registro predeterminado, si no se proporciona ninguno), se devuelven los metadatos de ese esquema. De lo contrario, se devuelve un 404 o un error. NotFound Las versiones del esquema con estado `Deleted` no se incluirán en los resultados.

Solicitud

- `SchemaId`: obligatorio: objeto SchemaId.

  Una estructura de encapsulador que contiene campos de identidad de esquema. La estructura contiene lo siguiente:

  - SchemaId$SchemaArn: el nombre de recurso de Amazon (ARN) del esquema. Uno de `SchemaArn` o `SchemaName` debe ser proporcionado.
  - SchemaId$SchemaName: el nombre del esquema. Uno de `SchemaArn` o `SchemaName` debe ser proporcionado.
- `SchemaDefinition` – Obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 170 000 bytes de largo como máximo, que coincide con el Custom string pattern #32.

  Definición del esquema para el que se requieren detalles de esquema.

Respuesta

- `SchemaVersionId`: cadena UTF-8, con 36 bytes de largo como mínimo y 36 bytes de largo como máximo, que coincide con el Custom string pattern #17.

  El ID de esquema de la versión del esquema.
- `SchemaArn`: cadena UTF-8, con 1 byte de largo como mínimo y 10 240 bytes de largo como máximo, que coincide con el Custom string pattern #22.

  Nombre de recurso de Amazon (ARN) del esquema.
- `DataFormat`: cadena UTF-8 (valores válidos: `AVRO` | `JSON` | `PROTOBUF`).

  El formato de datos de la definición de esquema. `AVRO`, `JSON` y `PROTOBUF` se admiten en la actualidad.
- `Status`: cadena UTF-8 (valores válidos: `AVAILABLE` | `PENDING` | `FAILURE` | `DELETING`).

El estado de la versión del esquema.

- `CreatedTime`: cadena UTF-8.

  La fecha y hora en que se creó el esquema.

Errores

- `InvalidInputException`
- `AccessDeniedException`
- `EntityNotFoundException`
- `InternalServiceException`

GetRegistry acción (Python: get_registry)

Describe el registro especificado en detalle.

Solicitud

- `RegistryId`: obligatorio: objeto RegistryId.

  Una estructura de encapsulador que puede contener el nombre del registro y el nombre de recurso de Amazon (ARN).

Respuesta

- `RegistryName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Custom string pattern #18.

  El nombre del registro.
- `RegistryArn`: cadena UTF-8, con 1 byte de largo como mínimo y 10 240 bytes de largo como máximo, que coincide con el Custom string pattern #22.

  Nombre de recurso de Amazon (ARN) del registro.
- `Description`: cadena de descripción de un máximo de 2048 bytes de largo, que coincide con el URI address multi-line string pattern.

  Es una descripción del recurso.

- `Status`: cadena UTF-8 (valores válidos: `AVAILABLE` | `DELETING`).

  Estado del registro.
- `CreatedTime`: cadena UTF-8.

  La fecha y hora en que se creó el registro.
- `UpdatedTime`: cadena UTF-8.

  La fecha y hora en que se actualizó el registro.

Errores

- `InvalidInputException`
- `AccessDeniedException`
- `EntityNotFoundException`
- `InternalServiceException`

PutSchemaVersionMetadata acción (Python: put_schema_version_metadata)

Coloca el par valor-clave de los metadatos para un ID de versión de esquema especificado. Se permitirá un máximo de 10 pares valor-clave por versión de esquema. Se pueden agregar a través de una o más llamadas.

Solicitud

- `SchemaId`: un objeto SchemaId.

  El ID único del esquema.
- `SchemaVersionNumber`: un objeto SchemaVersionNumber.

  El número de versión del esquema.
- `SchemaVersionId`: cadena UTF-8, con 36 bytes de largo como mínimo y 36 bytes de largo como máximo, que coincide con el Custom string pattern #17.

  El ID de versión único de la versión del esquema.
- `MetadataKeyValue`: obligatorio: objeto MetadataKeyValuePair.

Valor correspondiente de una clave de metadatos.

Respuesta

- `SchemaArn`: cadena UTF-8, con 1 byte de largo como mínimo y 10 240 bytes de largo como máximo, que coincide con el Custom string pattern #22.

  Nombre de recurso de Amazon (ARN) del esquema.

- `SchemaName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Custom string pattern #18.

  El nombre para el esquema.

- `RegistryName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Custom string pattern #18.

  El nombre para el registro.

- `LatestVersion`: booleano.

  La última versión del esquema.

- `VersionNumber`: número (largo) que no es inferior a 1 ni es superior a 100 000.

  El número de versión del esquema.

- `SchemaVersionId`: cadena UTF-8, con 36 bytes de largo como mínimo y 36 bytes de largo como máximo, que coincide con el Custom string pattern #17.

  El ID de versión único de la versión del esquema.

- `MetadataKey`: cadena UTF-8, con 1 byte de largo como mínimo y 128 bytes de largo como máximo, que coincide con el Custom string pattern #33.

  La clave de metadatos.

- `MetadataValue`: cadena UTF-8, con 1 byte de largo como mínimo y 256 bytes de largo como máximo, que coincide con el Custom string pattern #33.

  El valor de la clave de metadatos.

Errores

- `InvalidInputException`
- `AccessDeniedException`
- `AlreadyExistsException`
- `EntityNotFoundException`
- `ResourceNumberLimitExceededException`

QuerySchemaVersionMetadata acción (Python: query_schema_version_metadata)

Consulta la información de metadatos de versión del esquema.

Solicitud

- `SchemaId`: un objeto SchemaId.

  Una estructura de encapsulador que puede contener el nombre del esquema y el nombre de recurso de Amazon (ARN).
- `SchemaVersionNumber`: un objeto SchemaVersionNumber.

  El número de versión del esquema.
- `SchemaVersionId`: cadena UTF-8, con 36 bytes de largo como mínimo y 36 bytes de largo como máximo, que coincide con el Custom string pattern #17.

  El ID de versión único de la versión del esquema.
- `MetadataList`: matriz de objetos MetadataKeyValuePair.

  Busca los pares clave-valor para los metadatos, si no se proporcionan se obtendrá toda la información de los metadatos.
- `MaxResults`: número (entero) que no es inferior a 1 ni es superior a 50.

  Número máximo de resultados requeridos por página. Si no se proporciona el valor, el valor predeterminado será 25 por página.
- `NextToken`: cadena UTF-8.

  Token de continuación si se trata de una llamada de continuidad.

Respuesta

- `MetadataInfoMap`: matriz de mapas de pares clave-valor.

  Cada clave es una cadena UTF-8 con 1 byte de largo como mínimo y 128 bytes de largo como máximo, que coincide con Custom string pattern #33.

  Cada valor es un objeto A MetadataInfo.

  Mapa de una clave de metadatos y valores asociados.
- `SchemaVersionId`: cadena UTF-8, con 36 bytes de largo como mínimo y 36 bytes de largo como máximo, que coincide con el Custom string pattern #17.

  El ID de versión único de la versión del esquema.
- `NextToken`: cadena UTF-8.

  Token de continuación para paginar la lista de tokens obtenida; se devuelve si el segmento actual de la lista no es el último.

Errores

- `InvalidInputException`
- `AccessDeniedException`
- `EntityNotFoundException`

RemoveSchemaVersionMetadata acción (Python: remove_schema_version_metadata)

Elimina un par valor-clave de los metadatos de la versión del esquema para el ID de versión de esquema especificado.

Solicitud

- `SchemaId`: un objeto SchemaId.

  Una estructura de encapsulador que puede contener el nombre del esquema y el nombre de recurso de Amazon (ARN).
- `SchemaVersionNumber`: un objeto SchemaVersionNumber.

El número de versión del esquema.

- `SchemaVersionId`: cadena UTF-8, con 36 bytes de largo como mínimo y 36 bytes de largo como máximo, que coincide con el Custom string pattern #17.

  El ID de versión único de la versión del esquema.

- `MetadataKeyValue`: obligatorio: objeto MetadataKeyValuePair.

  El valor de la clave de metadatos.

Respuesta

- `SchemaArn`: cadena UTF-8, con 1 byte de largo como mínimo y 10 240 bytes de largo como máximo, que coincide con el Custom string pattern #22.

  Nombre de recurso de Amazon (ARN) del esquema.

- `SchemaName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Custom string pattern #18.

  El nombre del esquema.

- `RegistryName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Custom string pattern #18.

  El nombre del registro.

- `LatestVersion`: booleano.

  La última versión del esquema.

- `VersionNumber`: número (largo) que no es inferior a 1 ni es superior a 100 000.

  El número de versión del esquema.

- `SchemaVersionId`: cadena UTF-8, con 36 bytes de largo como mínimo y 36 bytes de largo como máximo, que coincide con el Custom string pattern #17.

  El ID de versión de la versión del esquema.

- `MetadataKey`: cadena UTF-8, con 1 byte de largo como mínimo y 128 bytes de largo como máximo, que coincide con el Custom string pattern #33.

  La clave de metadatos.

- `MetadataValue`: cadena UTF-8, con 1 byte de largo como mínimo y 256 bytes de largo como máximo, que coincide con el Custom string pattern #33.

  El valor de la clave de metadatos.

Errores

- `InvalidInputException`
- `AccessDeniedException`
- `EntityNotFoundException`

DeleteRegistry acción (Python: delete_registry)

Elimina todo el registro, incluido el esquema y todas sus versiones. Para obtener el estado de la operación de eliminación, puede llamar a la API `GetRegistry` después de la llamada asíncrona. Al eliminar un registro se desactivarán todas las operaciones en línea para el registro, como las API `UpdateRegistry`, `CreateSchema`, `UpdateSchema` y `RegisterSchemaVersion`.

Solicitud

- `RegistryId`: obligatorio: objeto RegistryId.

  Una estructura de encapsulador que puede contener el nombre del registro y el nombre de recurso de Amazon (ARN).

Respuesta

- `RegistryName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Custom string pattern #18.

  El nombre del registro que se está eliminando.
- `RegistryArn`: cadena UTF-8, con 1 byte de largo como mínimo y 10 240 bytes de largo como máximo, que coincide con el Custom string pattern #22.

  El nombre de recurso de Amazon (ARN) del registro que se está eliminando.
- `Status`: cadena UTF-8 (valores válidos: `AVAILABLE` | `DELETING`).

  Estado del registro. Una operación exitosa devolverá el estado `Deleting`.

Errores

- `InvalidInputException`
- `EntityNotFoundException`
- `AccessDeniedException`
- `ConcurrentModificationException`

DeleteSchema acción (Python: delete_schema)

Elimina todo el conjunto de esquemas, incluido el conjunto de esquemas y todas sus versiones. Para obtener el estado de la operación de eliminación, puede llamar a la API `GetSchema` después de la llamada asíncrona. Al eliminar un registro se desactivarán todas las operaciones en línea para el esquema, como las API `GetSchemaByDefinition` y `RegisterSchemaVersion`.

Solicitud

- `SchemaId`: obligatorio: objeto SchemaId.

  Es una estructura de encapsulador que puede contener el nombre del esquema y el nombre de recurso de Amazon (ARN).

Respuesta

- `SchemaArn`: cadena UTF-8, con 1 byte de largo como mínimo y 10 240 bytes de largo como máximo, que coincide con el Custom string pattern #22.

  Nombre de recurso de Amazon (ARN) del esquema que se está eliminando.
- `SchemaName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Custom string pattern #18.

  El nombre del esquema que se está eliminando.
- `Status`: cadena UTF-8 (valores válidos: `AVAILABLE` | `PENDING` | `DELETING`).

  Estado del esquema.

Errores

- `InvalidInputException`

- `EntityNotFoundException`
- `AccessDeniedException`
- `ConcurrentModificationException`

DeleteSchemaVersions acción (Python: delete_schema_versions)

Elimina versiones del esquema especificado. Se puede suministrar un número o rango de versiones. Si el modo de compatibilidad prohíbe eliminar una versión necesaria, como HACIA ATRÁS_COMPLETO, se devuelve un error. Llamar a la API `GetSchemaVersions` después de esta llamada arrojará una lista del estado de las versiones eliminadas.

Cuando el rango de números de versión contiene versiones de punto de comprobación, la API devolverá un conflicto 409 y no continuará con la eliminación. Primero debe eliminar el punto de comprobación con la API `DeleteSchemaCheckpoint`, antes de usar esta API.

No puede usar la API `DeleteSchemaVersions` para eliminar la primera versión del esquema de un conjunto de esquemas. La primera versión del esquema sólo puede ser eliminada por la API `DeleteSchema`. Esta operación también eliminará los `SchemaVersionMetadata` asociados con las versiones del esquema. Se aplicarán eliminaciones fragmentadas en la base de datos.

Si el modo de compatibilidad prohíbe eliminar una versión necesaria, como HACIA ATRÁS_COMPLETO, se devuelve un error.

Solicitud

- `SchemaId`: obligatorio: objeto SchemaId.

  Es una estructura de encapsulador que puede contener el nombre del esquema y el nombre de recurso de Amazon (ARN).

- `Versions` – Obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 100 000 bytes de largo como máximo, que coincide con el Custom string pattern #34.

  Puede suministrarse un rango de versiones que puede tener el siguiente formato:

  - un único número de versión, 5
  - un rango, 5-8: elimina las versiones 5, 6, 7, 8

Respuesta

- `SchemaVersionErrors`: matriz de objetos SchemaVersionErrorItem.

  Una lista de objetos `SchemaVersionErrorItem`, cada uno de los cuales contiene un error y una versión de esquema.

Errores

- `InvalidInputException`
- `EntityNotFoundException`
- `AccessDeniedException`
- `ConcurrentModificationException`

Flujos de trabajo

La API de flujos de trabajo permite describir los tipos de datos y la API relacionados con la creación, actualización o visualización de flujos de trabajo en AWS Glue. El historial de ejecución de trabajos está disponible durante 90 días para su flujo de trabajo y ejecución de trabajos.

Tipos de datos

- Estructura JobNodeDetails
- Estructura CrawlerNodeDetails
- Estructura TriggerNodeDetails
- Estructura de rastreo
- Estructura de nodos
- Estructura perimetral
- Estructura de flujo de trabajo
- Estructura WorkflowGraph
- Estructura WorkflowRun
- Estructura WorkflowRunStatistics
- Estructura StartingEventBatchCondition

- Estructura de proyecto
- Estructura BlueprintDetails
- Estructura LastActiveDefinition
- Estructura BlueprintRun

Estructura JobNodeDetails

Los detalles de un nodo de trabajo presente en el flujo de trabajo.

Campos

- `JobRuns`: matriz de objetos JobRun.

  La información para las ejecuciones de trabajo representada por el nodo de trabajo.

Estructura CrawlerNodeDetails

Los detalles de un nodo de rastreador presente en el flujo de trabajo.

Campos

- `Crawls`: matriz de objetos Rastreo.

  Una lista de rastreadores representada por el nodo de rastreo.

Estructura TriggerNodeDetails

Los detalles de un nodo de desencadenador presente en el flujo de trabajo.

Campos

- `Trigger`: un objeto Desencadenador.

  La información del desencadenador representado por el nodo de desencadenador.

Estructura de rastreo

Los detalles de un rastreo en el flujo de trabajo.

Campos

- `State`: cadena UTF-8 (valores válidos: `RUNNING` | `CANCELLING` | `CANCELLED` | `SUCCEEDED` | `FAILED` | `ERROR`).

  El estado del rastreador.
- `StartedOn`: marca temporal.

  La fecha y hora en las que se inició el rastreo.
- `CompletedOn`: marca temporal.

  La fecha y hora en las que se completó el rastreo.
- `ErrorMessage`: cadena de descripción de un máximo de 2048 bytes de largo, que coincide con el URI address multi-line string pattern.

  El mensaje de error asociado con el rastreo.
- `LogGroup`: cadena UTF-8, con 1 byte de largo como mínimo y 512 bytes de largo como máximo, que coincide con el Log group string pattern.

  El grupo de registros asociado al rastreo.
- `LogStream`: cadena UTF-8, con 1 byte de largo como mínimo y 512 bytes de largo como máximo, que coincide con el Log-stream string pattern.

  El flujo de registros asociado al rastreo.

Estructura de nodos

Un nodo representa un componente de AWS Glue (desencadenador, rastreador o trabajo) en un gráfico de flujo de trabajo.

Campos

- `Type`: cadena UTF-8 (valores válidos: `CRAWLER` | `JOB` | `TRIGGER`).

  El tipo de componente de AWS Glue representado por el nodo.
- `Name`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre del componente de AWS Glue representado por el nodo.

- `UniqueId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID exclusivo asignado al nodo en el flujo de trabajo.

- `TriggerDetails`: un objeto TriggerNodeDetails.

  Detalles del desencadenador cuando el nodo representa a un desencadenador.

- `JobDetails`: un objeto JobNodeDetails.

  Detalles del trabajo cuando el nodo representa a un trabajo.

- `CrawlerDetails`: un objeto CrawlerNodeDetails.

  Detalles del rastreador cuando el nodo representa a un rastreador.

Estructura perimetral

Un borde representa una conexión dirigida entre dos componentes de AWS Glue que forman parte del flujo de trabajo al que pertenece el borde.

Campos

- `SourceId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nodo único dentro del flujo de trabajo en el que se inicia el perímetro.

- `DestinationId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El único del nodo en el flujo de trabajo donde finaliza el perímetro.

Estructura de flujo de trabajo

Un flujo de trabajo es una recopilación de varios trabajos y rastreadores de AWS Glue que se ejecutan para completar una tarea ETL compleja. Los flujos de trabajo administran la ejecución y monitoreo de todos sus trabajos y rastreadores.

Campos

- `Name`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre del flujo de trabajo.
- `Description`: cadena UTF-8.

  Una descripción del flujo de trabajo.
- `DefaultRunProperties`: matriz de mapas de pares clave-valor.

  Cada clave es una cadena UTF-8 con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Cada valor es una cadena UTF-8.

  Una colección de propiedades que se va a utilizar como parte de cada ejecución del flujo de trabajo. Las propiedades de ejecución están disponibles para cada trabajo del flujo de trabajo. Un trabajo puede modificar las propiedades de los siguientes trabajos en el flujo.
- `CreatedOn`: marca temporal.

  La fecha y hora en las que se ha creado el flujo de trabajo.
- `LastModifiedOn`: marca temporal.

  La fecha y hora en las que se ha modificado el flujo de trabajo por última vez.
- `LastRun`: un objeto WorkflowRun.

  La información sobre la última ejecución del flujo de trabajo.
- `Graph`: un objeto WorkflowGraph.

  El gráfico que representa todos los componentes de AWS Glue que pertenecen al flujo de trabajo como nodos y conexiones dirigidas entre ellos como bordes.
- `CreationStatus`: cadena UTF-8 (valores válidos: `CREATING` | `CREATED` | `CREATION_FAILED`).

  El estado de creación del flujo de trabajo.
- `MaxConcurrentRuns`: número (entero).

  Puede utilizar este parámetro para evitar varias actualizaciones no deseadas de datos, para controlar los costos o, en algunos casos, para evitar que se supere el número máximo de

ejecuciones concurrentes de cualquiera de los trabajos de los componentes. Si deja este parámetro en blanco, no hay límite en el número de ejecuciones concurrentes de flujos de trabajo.

- `BlueprintDetails`: un objeto BlueprintDetails.

  Esta estructura indica los detalles del proyecto desde el que se crea este flujo de trabajo en particular.

Estructura WorkflowGraph

Un gráfico de flujo de trabajo representa el flujo de trabajo completo que contiene todos los componentes de AWS Glue presentes en el flujo de trabajo y todas las conexiones dirigidas entre ellos.

Campos

- `Nodes`: matriz de objetos Nodo.

  Una lista de los componentes de AWS Glue que pertenecen al flujo de trabajo representados como nodos.

- `Edges`: matriz de objetos Periferia.

  Una lista de todas las conexiones dirigidas entre los nodos que pertenecen al flujo de trabajo.

Estructura WorkflowRun

Una ejecución de flujo de trabajo proporciona toda la información sobre el tiempo de ejecución.

Campos

- `Name`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre del flujo de trabajo que se ejecutó.

- `WorkflowRunId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID de esta ejecución de flujo de trabajo.

- `PreviousRunId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID de la ejecución de flujo de trabajo anterior.

- `WorkflowRunProperties`: matriz de mapas de pares clave-valor.

  Cada clave es una cadena UTF-8 con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Cada valor es una cadena UTF-8.

  Las propiedades de ejecución de flujo de trabajo que se establecieron durante la ejecución.

- `StartedOn`: marca temporal.

  La fecha y hora en las que se inició la ejecución de flujo de trabajo.

- `CompletedOn`: marca temporal.

  La fecha y hora en las que se completó la ejecución de flujo de trabajo.

- `Status`: cadena UTF-8 (valores válidos: `RUNNING` | `COMPLETED` | `STOPPING` | `STOPPED` | `ERROR`).

  El estado de la ejecución de flujo de trabajo.

- `ErrorMessage`: cadena UTF-8.

  Este mensaje de error describe cualquier error que se haya producido al iniciar la ejecución del flujo de trabajo. Actualmente, el único mensaje de error es “Concurrent runs exceeded for workflow: `foo` (Exceso de ejecuciones simultáneas para el flujo de trabajo: foo)”.

- `Statistics`: un objeto WorkflowRunStatistics.

  Las estadísticas de la ejecución.

- `Graph`: un objeto WorkflowGraph.

  El gráfico que representa todos los componentes de AWS Glue que pertenecen al flujo de trabajo como nodos y conexiones dirigidas entre ellos como bordes.

- `StartingEventBatchCondition`: un objeto StartingEventBatchCondition.

  La condición del lote que inició la ejecución del flujo de trabajo.

Estructura WorkflowRunStatistics

Las estadísticas de ejecución de flujo de trabajo proporcionan estadísticas sobre la ejecución de flujo de trabajo.

Campos

- `TotalActions`: número (entero).

  Número total de acciones en la ejecución de flujo de trabajo.

- `TimeoutActions`: número (entero).

  Número total de acciones para las que se agotó el tiempo de espera.

- `FailedActions`: número (entero).

  Número total de acciones erróneas.

- `StoppedActions`: número (entero).

  Número total de acciones detenidas.

- `SucceededActions`: número (entero).

  Número total de acciones correctas.

- `RunningActions`: número (entero).

  Número total de acciones en estado de ejecución.

- `ErroredActions`: número (entero).

  Indica el recuento de ejecuciones de trabajos en estado ERROR en la ejecución del flujo de trabajo.

- `WaitingActions`: número (entero).

  Indica el recuento de ejecuciones de trabajos en estado WAITING (EN ESPERA) en la ejecución del flujo de trabajo.

Estructura StartingEventBatchCondition

La condición del lote que inició la ejecución del flujo de trabajo. El número de eventos indicado en el tamaño del lote llegó, en cuyo caso el miembro BatchSize es distinto de cero, o la ventana del lote venció, en cuyo caso el miembro BatchWindow es distinto de cero.

Campos

- `BatchSize`: número (entero).

  Número de eventos en el lote.
- `BatchWindow`: número (entero).

  Duración en segundos de la ventana del lote.

Estructura de proyecto

Los detalles de un proyecto.

Campos

- `Name`: cadena UTF-8, con 1 byte de largo como mínimo y 128 bytes de largo como máximo, que coincide con el Custom string pattern #27.

  El nombre del proyecto.
- `Description`: cadena UTF-8, con 1 byte de largo como mínimo y 512 bytes de largo como máximo.

  Descripción del proyecto.
- `CreatedOn`: marca temporal.

  La fecha y la hora en que se registró el proyecto.
- `LastModifiedOn`: marca temporal.

  La fecha y hora en la que se modificó el proyecto por última vez.
- `ParameterSpec`: cadena UTF-8, con 1 byte de largo como mínimo y 131 072 bytes de largo como máximo.

  Cadena JSON que indica la lista de especificaciones de parámetros para el proyecto.

- `BlueprintLocation`: cadena UTF-8.

  Especifica la ruta en Amazon S3 donde se publica el proyecto.

- `BlueprintServiceLocation`: cadena UTF-8.

  Especifica una ruta en Amazon S3 donde se copia el proyecto cuando se llama a `CreateBlueprint/UpdateBlueprint` para registrar el proyecto en AWS Glue.

- `Status`: cadena UTF-8 (valores válidos: `CREATING` | `ACTIVE` | `UPDATING` | `FAILED`).

  El estado del registro del proyecto.

  - Creating (Creación): el registro del proyecto está en curso.
  - Active (Activo): el proyecto se ha registrado correctamente.
  - Updating (Actualización): está en curso una actualización al registro del proyecto.
  - Failed (Error): error en el registro del proyecto.

- `ErrorMessage`: cadena UTF-8.

  Mensaje de error.

- `LastActiveDefinition`: un objeto LastActiveDefinition.

  Cuando hay varias versiones de un proyecto y la última versión tiene algunos errores, este atributo indica la última definición del proyecto correcta que está disponible con el servicio.

Estructura BlueprintDetails

Los detalles de un proyecto.

Campos

- `BlueprintName`: cadena UTF-8, con 1 byte de largo como mínimo y 128 bytes de largo como máximo, que coincide con el Custom string pattern #27.

  El nombre del proyecto.

- `RunId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID de ejecución de este proyecto.

Estructura LastActiveDefinition

Cuando hay varias versiones de un proyecto y la última versión tiene algunos errores, este atributo indica la última definición del proyecto correcta que está disponible con el servicio.

Campos

- `Description`: cadena UTF-8, con 1 byte de largo como mínimo y 512 bytes de largo como máximo.

  Descripción del proyecto.
- `LastModifiedOn`: marca temporal.

  La fecha y hora en la que se modificó el proyecto por última vez.
- `ParameterSpec`: cadena UTF-8, con 1 byte de largo como mínimo y 131 072 bytes de largo como máximo.

  Una cadena JSON que especifica los parámetros para el proyecto.
- `BlueprintLocation`: cadena UTF-8.

  Especifica una ruta en Amazon S3 donde el desarrollador de AWS Glue publica el proyecto.
- `BlueprintServiceLocation`: cadena UTF-8.

  Especifica una ruta en Amazon S3 donde se copia el esquema cuando se crea o actualiza el esquema.

Estructura BlueprintRun

Los detalles de una ejecución del proyecto.

Campos

- `BlueprintName`: cadena UTF-8, con 1 byte de largo como mínimo y 128 bytes de largo como máximo, que coincide con el Custom string pattern #27.

  El nombre del proyecto.
- `RunId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

El ID de ejecución de esta ejecución del proyecto.

- `WorkflowName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre de un flujo de trabajo que se crea como resultado de una ejecución correcta del proyecto. Si la ejecución del proyecto presenta un error, no se creará ningún flujo de trabajo.

- `State`: cadena UTF-8 (valores válidos: `RUNNING` | `SUCCEEDED` | `FAILED` | `ROLLING_BACK`).

  Estado de la ejecución del proyecto. Los valores posibles son los siguientes:

  - Running (Ejecución): la ejecución del proyecto está en curso.
  - Succeeded (Correcta): la ejecución del proyecto se completó correctamente.
  - Failed (Error): la ejecución del proyecto falló y se completó la restauración.
  - Rolling back (Restauración): la ejecución del proyecto falló y la restauración está en curso.

- `StartedOn`: marca temporal.

  La fecha y la hora en las que se inició la ejecución del proyecto.

- `CompletedOn`: marca temporal.

  La fecha y la hora en las que se completó la ejecución de proyecto.

- `ErrorMessage`: cadena UTF-8.

  Indica los errores que se detectan al ejecutar el proyecto.

- `RollbackErrorMessage`: cadena UTF-8.

  Si hay algún error al crear las entidades de un flujo de trabajo, intentamos restaurar las entidades creadas hasta ese punto y eliminarlas. Este atributo indica los errores detectados al intentar eliminar las entidades que se crean.

- `Parameters`: cadena UTF-8, con 1 byte de largo como mínimo y 131 072 bytes de largo como máximo.

  Los parámetros del proyecto como una cadena. Será necesario que proporcione un valor para cada clave que se requiera a partir de la especificación de parámetros definida en la `Blueprint $ParameterSpec`.

- `RoleArn`: cadena UTF-8, con 1 byte como mínimo o más de 1024 bytes de largo, que coincide con el Custom string pattern #26.

El ARN del rol. Este rol será asumido por el servicio de AWS Glue y se utilizará para crear el flujo de trabajo y otras entidades de un flujo de trabajo.

Operaciones

- Acción CreateWorkflow (Python: create_workflow)
- Acción UpdateWorkflow (Python: update_workflow)
- DeleteWorkflow Action (Python: delete_workflow)
- Acción GetWorkFlow (Python: get_workflow)
- Acción ListWorkflows (Python: list_workflows)
- Acción BatchGetWorkflows (Python: batch_get_workflows)
- Acción GetWorkflowRun (Python: get_workflow_run)
- Acción GetWorkflowRuns (Python: get_workflow_runs)
- Acción GetWorkflowRunProperties (Python: get_workflow_run_properties)
- PutWorkflowRunProperties Action (Python: put_workflow_run_properties)
- Acción CreateBlueprint (Python: create_blueprint)
- Acción UpdateBlueprint (Python: update_blueprint)
- Acción DeleteBlueprint (Python: delete_blueprint)
- Acción ListBlueprints (Python: list_blueprints)
- Acción BatchGetBlueprints (Python: batch_get_blueprints)
- Acción StartBlueprintRun (Python: start_blueprint_run)
- Acción GetBlueprintRun (Python: get_blueprint_run)
- Acción GetBlueprintRuns (Python: get_blueprint_runs)
- Acción StartWorkflowRun (Python: start_workflow_run)
- Acción StopWorkflowRun (Python: stop_workflow_run)
- Acción ResumeWorkflowRun (Python: resume_workflow_run)

Acción CreateWorkflow (Python: create_workflow)

Crea un nuevo flujo de trabajo.

Solicitud

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre que se va a asignar al flujo de trabajo. Debe ser único en su cuenta.

- `Description`: cadena UTF-8.

  Una descripción del flujo de trabajo.

- `DefaultRunProperties`: matriz de mapas de pares clave-valor.

  Cada clave es una cadena UTF-8 con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Cada valor es una cadena UTF-8.

  Una colección de propiedades que se va a utilizar como parte de cada ejecución del flujo de trabajo.

- `Tags`: matriz de mapas de pares de clave-valor, con 50 pares como máximo.

  Cada clave es una cadena UTF-8 con una longitud de entre 1 y 128 bytes.

  Cada valor es una cadena UTF-8 que no tiene más de 256 bytes de largo.

  Las etiquetas que se van a utilizar con este flujo de trabajo.

- `MaxConcurrentRuns`: número (entero).

  Puede utilizar este parámetro para evitar varias actualizaciones no deseadas de datos, para controlar los costos o, en algunos casos, para evitar que se supere el número máximo de ejecuciones concurrentes de cualquiera de los trabajos de los componentes. Si deja este parámetro en blanco, no hay límite en el número de ejecuciones concurrentes de flujos de trabajo.

Respuesta

- `Name`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre del flujo de trabajo que se proporcionó como parte de la solicitud.

Errores

- `AlreadyExistsException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`
- `ConcurrentModificationException`

Acción UpdateWorkflow (Python: update_workflow)

Actualiza un flujo de trabajo existente.

Solicitud

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre del flujo de trabajo que se va a actualizar.
- `Description`: cadena UTF-8.

  La descripción del flujo de trabajo.
- `DefaultRunProperties`: matriz de mapas de pares clave-valor.

  Cada clave es una cadena UTF-8 con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Cada valor es una cadena UTF-8.

  Una colección de propiedades que se va a utilizar como parte de cada ejecución del flujo de trabajo.
- `MaxConcurrentRuns`: número (entero).

  Puede utilizar este parámetro para evitar varias actualizaciones no deseadas de datos, para controlar los costos o, en algunos casos, para evitar que se supere el número máximo de ejecuciones concurrentes de cualquiera de los trabajos de los componentes. Si deja este parámetro en blanco, no hay límite en el número de ejecuciones concurrentes de flujos de trabajo.

Respuesta

- Name: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre del flujo de trabajo que se especificó en la entrada.

Errores

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `ConcurrentModificationException`

DeleteWorkflow Action (Python: delete_workflow)

Elimina un flujo de trabajo.

Solicitud

- Name: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre del flujo de trabajo que se va a eliminar.

Respuesta

- Name: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre del flujo de trabajo especificado en la entrada.

Errores

- `InvalidInputException`
- `InternalServiceException`

- `OperationTimeoutException`
- `ConcurrentModificationException`

Acción GetWorkFlow (Python: get_workflow)

Recupera metadatos de recursos para un flujo de trabajo.

Solicitud

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre del flujo de trabajo que se va a recuperar.
- `IncludeGraph`: booleano.

  Especifica si se debe incluir un gráfico cuando se devuelven los metadatos de recursos de flujo de trabajo.

Respuesta

- `Workflow`: un objeto Flujo de trabajo.

  Metadatos de recursos para el flujo de trabajo.

Errores

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`

Acción ListWorkflows (Python: list_workflows)

Permite enumerar los nombres de flujos de trabajo creados en la cuenta.

Solicitud

- `NextToken`: cadena UTF-8.

Token de continuación, si se trata de una solicitud de continuidad.

- `MaxResults`: número (entero), mayor que 1 y menor que 25.

  Tamaño máximo de una lista que se devolverá.

Respuesta

- `Workflows`: matriz de cadenas UTF-8, con una cadena como mínimo y 25 cadenas como máximo.

  Lista de nombres de flujos de trabajo en la cuenta.
- `NextToken`: cadena UTF-8.

  Token de continuación, si no se han devuelto todos los nombres de flujo de trabajo.

Errores

- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

Acción BatchGetWorkflows (Python: batch_get_workflows)

Devuelve la lista de metadatos de recursos de una determinada lista de nombres de flujos de trabajo. Después de llamar a la operación `ListWorkflows`, puede llamar a esta operación para obtener acceso a los datos a los que ha concedido permisos. Esta operación admite todos los permisos de IAM, incluidas las condiciones de permisos que utilizan etiquetas.

Solicitud

- `Names` – Obligatorio: matriz de cadenas UTF-8, con una cadena como mínimo y 25 cadenas como máximo.

  Lista de nombres de flujos de trabajo, que pueden ser los nombres devueltos en la operación `ListWorkflows`.
- `IncludeGraph`: booleano.

Especifica si se debe incluir un gráfico cuando se devuelven los metadatos de recursos de flujo de trabajo.

Respuesta

- `Workflows`: matriz de objetos Flujo de trabajo, con una estructura como mínimo y 25 estructuras como máximo.

  Una lista de los metadatos de recursos de flujo de trabajo.
- `MissingWorkflows`: matriz de cadenas UTF-8, con una cadena como mínimo y 25 cadenas como máximo.

  Una lista de nombres de flujos de trabajo no encontrados.

Errores

- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`

Acción GetWorkflowRun (Python: get_workflow_run)

Recupera los metadatos para una ejecución de flujo de trabajo especificada. El historial de ejecución de trabajos está disponible durante 90 días para su flujo de trabajo y ejecución de trabajos.

Solicitud

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre del flujo de trabajo en ejecución.
- `RunId`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID de la ejecución de flujos de trabajo.
- `IncludeGraph`: booleano.

Especifica si se debe incluir el gráfico de flujo de trabajo en la respuesta o no.

Respuesta

- `Run`: un objeto WorkflowRun.

  Los metadatos de ejecución de ejecución de flujo de trabajo solicitados.

Errores

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`

Acción GetWorkflowRuns (Python: get_workflow_runs)

Recupera los metadatos para todas las ejecuciones de un flujo de trabajo especificado.

Solicitud

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre del flujo de trabajo cuyos metadatos de ejecuciones deben devolverse.
- `IncludeGraph`: booleano.

  Especifica si se debe incluir el gráfico de flujo de trabajo en la respuesta o no.
- `NextToken`: cadena UTF-8.

  Tamaño máximo de la respuesta.
- `MaxResults`: número (entero) que no es inferior a 1 ni es superior a 1000.

  El número máximo de ejecuciones de flujo de trabajo que se incluirá en la respuesta.

Respuesta

- `Runs`: matriz de objetos WorkflowRun, con una estructura como mínimo y 1000 estructuras como máximo.

  Una lista de objetos de metadatos de ejecución de flujo de trabajo.
- `NextToken`: cadena UTF-8.

  Token de continuación, si no se han devuelto todas las ejecuciones de flujo de trabajo solicitadas.

Errores

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`

Acción GetWorkflowRunProperties (Python: get_workflow_run_properties)

Permite recuperar las propiedades de ejecución de flujo de trabajo que se establecieron durante la ejecución.

Solicitud

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre del flujo de trabajo que se ejecutó.
- `RunId`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID de la ejecución de flujo de trabajo cuyas propiedades de ejecución se deben devolver.

Respuesta

- `RunProperties`: matriz de mapas de pares clave-valor.

Cada clave es una cadena UTF-8 con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

Cada valor es una cadena UTF-8.

Las propiedades de ejecución de flujo de trabajo que se establecieron durante la ejecución especificada.

Errores

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`

PutWorkflowRunProperties Action (Python: put_workflow_run_properties)

Permite colocar las propiedades de ejecución del flujo de trabajo especificado para la ejecución del flujo de trabajo determinado. Si una propiedad ya existe para la ejecución especificada, sobrescribe el valor o añade la propiedad a las propiedades existentes.

Solicitud

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre del flujo de trabajo que se ejecutó.
- `RunId`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID de la ejecución de flujo de trabajo para el que se deben actualizar las propiedades de ejecución.
- `RunProperties` – Obligatorio: matriz de mapas de pares clave-valor.

  Cada clave es una cadena UTF-8 con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

Cada valor es una cadena UTF-8.

Las propiedades que colocar para la ejecución especificada.

Respuesta

- Sin parámetros de respuesta.

Errores

- `AlreadyExistsException`
- `EntityNotFoundException`
- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`
- `ConcurrentModificationException`

Acción CreateBlueprint (Python: create_blueprint)

Registra un proyecto con AWS Glue.

Solicitud

- `Name` – Obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 128 bytes de largo como máximo, que coincide con el Custom string pattern #27.

  El nombre del proyecto.
- `Description`: cadena UTF-8, con 1 byte de largo como mínimo y 512 bytes de largo como máximo.

  Una descripción del proyecto.
- `BlueprintLocation` – Obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 8192 bytes de largo como máximo, que coincide con el Custom string pattern #28.

  Especifica la ruta en Amazon S3 donde se publica el proyecto.

- `Tags`: matriz de mapas de pares de clave-valor, con 50 pares como máximo.

  Cada clave es una cadena UTF-8 con una longitud de entre 1 y 128 bytes.

  Cada valor es una cadena UTF-8 que no tiene más de 256 bytes de largo.

  Las etiquetas que se van a aplicar a este proyecto.

Respuesta

- `Name`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Devuelve el nombre del proyecto que se registró.

Errores

- `AlreadyExistsException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`
- `ResourceNumberLimitExceededException`

Acción UpdateBlueprint (Python: update_blueprint)

Actualiza un proyecto registrado.

Solicitud

- `Name` – Obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 128 bytes de largo como máximo, que coincide con el Custom string pattern #27.

  El nombre del proyecto.

- `Description`: cadena UTF-8, con 1 byte de largo como mínimo y 512 bytes de largo como máximo.

  Una descripción del proyecto.

- `BlueprintLocation` – Obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 8192 bytes de largo como máximo, que coincide con el Custom string pattern #28.

  Especifica la ruta en Amazon S3 donde se publica el proyecto.

Respuesta

- `Name`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Devuelve el nombre del proyecto que se actualizó.

Errores

- `EntityNotFoundException`
- `ConcurrentModificationException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`
- `IllegalBlueprintStateException`

Acción DeleteBlueprint (Python: delete_blueprint)

Elimina un proyecto existente.

Solicitud

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre del proyecto que se va a eliminar.

Respuesta

- `Name`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

Devuelve el nombre del proyecto que se eliminó.

Errores

- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

Acción ListBlueprints (Python: list_blueprints)

Muestra todos los nombres de proyecto de una cuenta.

Solicitud

- `NextToken`: cadena UTF-8.

  Token de continuación, si se trata de una solicitud de continuidad.
- `MaxResults`: número (entero), mayor que 1 y menor que 25.

  Tamaño máximo de una lista que se devolverá.
- `Tags`: matriz de mapas de pares de clave-valor, con 50 pares como máximo.

  Cada clave es una cadena UTF-8 con una longitud de entre 1 y 128 bytes.

  Cada valor es una cadena UTF-8 que no tiene más de 256 bytes de largo.

  Filtra la lista por una etiqueta de recursos de AWS.

Respuesta

- `Blueprints`: matriz de cadenas UTF-8.

  Lista de los nombres del proyectos en la cuenta.
- `NextToken`: cadena UTF-8.

  Un token de continuación, si no se han devuelto todos los nombres de proyectos.

Errores

- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

Acción BatchGetBlueprints (Python: batch_get_blueprints)

Recupera información sobre una lista de proyectos.

Solicitud

- `Names` – Obligatorio: matriz de cadenas UTF-8, con una cadena como mínimo y 25 cadenas como máximo.

  Una lista de nombres de proyectos.
- `IncludeBlueprint`: booleano.

  Especifica si se debe incluir el proyecto en la respuesta.
- `IncludeParameterSpec`: booleano.

  Especifica si se deben incluir los parámetros, como una cadena de JSON, para el proyecto en la respuesta.

Respuesta

- `Blueprints`: matriz de objetos Proyecto.

  Devuelve una lista de proyectos como un objeto `Blueprints`.
- `MissingBlueprints`: matriz de cadenas UTF-8.

  Devuelve una lista de `BlueprintNames` que no se encontraron.

Errores

- `InternalServiceException`
- `OperationTimeoutException`

- `InvalidInputException`

Acción StartBlueprintRun (Python: start_blueprint_run)

Inicia una nueva ejecución del proyecto especificado.

Solicitud

- `BlueprintName` – Obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 128 bytes de largo como máximo, que coincide con el Custom string pattern #27.

  El nombre del proyecto.
- `Parameters`: cadena UTF-8, con 1 byte de largo como mínimo y 131 072 bytes de largo como máximo.

  Especifica los parámetros como un objeto `BlueprintParameters`.
- `RoleArn` – Obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 1024 bytes de largo como máximo, que coincide con el Custom string pattern #26.

  Especifica el rol de IAM utilizado para crear el flujo de trabajo.

Respuesta

- `RunId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID de ejecución de esta ejecución del proyecto.

Errores

- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`
- `ResourceNumberLimitExceededException`
- `EntityNotFoundException`
- `IllegalBlueprintStateException`

Acción GetBlueprintRun (Python: get_blueprint_run)

Recupera los detalles de una ejecución de proyecto.

Solicitud

- `BlueprintName` – Obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 128 bytes de largo como máximo, que coincide con el Custom string pattern #27.

  El nombre del proyecto.
- `RunId`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID de ejecución de la ejecución de proyecto que desea recuperar.

Respuesta

- `BlueprintRun`: un objeto BlueprintRun.

  Devuelve un objeto `BlueprintRun`.

Errores

- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`

Acción GetBlueprintRuns (Python: get_blueprint_runs)

Recupera los detalles de las ejecuciones de proyecto de un proyecto especificado.

Solicitud

- `BlueprintName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre del proyecto.
- `NextToken`: cadena UTF-8.

Token de continuación, si se trata de una solicitud de continuidad.

- `MaxResults`: número (entero) que no es inferior a 1 ni es superior a 1000.

  Tamaño máximo de una lista que se devolverá.

Respuesta

- `BlueprintRuns`: matriz de objetos BlueprintRun.

  Devuelve una lista de objetos `BlueprintRun`.
- `NextToken`: cadena UTF-8.

  Un token de continuación, si no se han devuelto todas las ejecuciones de proyectos.

Errores

- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`

Acción StartWorkflowRun (Python: start_workflow_run)

Inicia una nueva ejecución del flujo de trabajo especificado.

Solicitud

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre del flujo de trabajo que se va a iniciar.
- `RunProperties`: matriz de mapas de pares clave-valor.

  Cada clave es una cadena UTF-8 con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Cada valor es una cadena UTF-8.

Las propiedades de ejecución del flujo de trabajo para la ejecución del flujo de trabajo nuevo.

Respuesta

- `RunId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Una identificación para la nueva carrera.

Errores

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`
- `ConcurrentRunsExceededException`

Acción StopWorkflowRun (Python: stop_workflow_run)

Detiene la ejecución del flujo de trabajo especificado.

Solicitud

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre del flujo de trabajo que se va a detener.
- `RunId`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID de los flujos de trabajo que se detendrán.

Respuesta

- Sin parámetros de respuesta.

Errores

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `IllegalWorkflowStateException`

Acción ResumeWorkflowRun (Python: resume_workflow_run)

Reinicia los nodos seleccionados de una ejecución de flujo de trabajo anterior parcialmente completada y reanuda la ejecución del flujo de trabajo. Se ejecutan los nodos seleccionados y todos los nodos descendentes a los nodos seleccionados.

Solicitud

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre del flujo de trabajo que se reanudará.
- `RunId`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID de la ejecución del flujo de trabajo que se reanudará.
- `NodeIds` – Obligatorio: una matriz de cadenas UTF-8.

  Una lista de ID de nodos para los nodos que desea reiniciar. Los nodos que se reiniciarán deben tener un intento de ejecución en la ejecución original.

Respuesta

- `RunId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nuevo ID asignado a la ejecución del flujo de trabajo reanudado. Cada reanudación de una ejecución de flujo de trabajo tendrá un nuevo ID de ejecución.
- `NodeIds`: matriz de cadenas UTF-8.

Una lista de los ID de los nodos para los nodos que realmente se reiniciaron.

Errores

- `InvalidInputException`
- `EntityNotFoundException`
- `InternalServiceException`
- `OperationTimeoutException`
- `ConcurrentRunsExceededException`
- `IllegalWorkflowStateException`

Perfiles de uso

La API de perfiles de uso describe los tipos de datos y la API relacionados con la creación, actualización o visualización de los perfiles de uso en AWS Glue.

Tipos de datos

- ProfileConfiguration estructura
- ConfigurationObject estructura
- UsageProfileDefinition estructura

ProfileConfiguration estructura

Especifica los valores de trabajo y sesión que un administrador configura en un perfil AWS Glue de uso.

Campos

- `SessionConfiguration`: matriz de mapas de pares clave-valor.

  Cada clave es una cadena UTF-8 con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Cada valor es un objeto A ConfigurationObject.

Un mapa de valores clave de los parámetros de configuración de las sesiones. AWS Glue

- `JobConfiguration`: matriz de mapas de pares clave-valor.

  Cada clave es una cadena UTF-8 con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Cada valor es un objeto A ConfigurationObject.

  Un mapa de valores clave de los parámetros de configuración de los trabajos. AWS Glue

ConfigurationObject estructura

Especifica los valores que un administrador establece para cada parámetro de trabajo o sesión configurado en un perfil de AWS Glue uso.

Campos

- `DefaultValue`: cadena UTF-8, con 1 byte de largo como mínimo y 128 bytes de largo como máximo, que coincide con el Custom string pattern #31.

  Un valor predeterminado para el parámetro.
- `AllowedValues`: matriz de cadenas UTF-8.

  Una lista de valores permitidos para el parámetro.
- `MinValue`: cadena UTF-8, con 1 byte de largo como mínimo y 128 bytes de largo como máximo, que coincide con el Custom string pattern #31.

  Un valor mínimo permitido para el parámetro.
- `MaxValue`: cadena UTF-8, con 1 byte de largo como mínimo y 128 bytes de largo como máximo, que coincide con el Custom string pattern #31.

  Un valor máximo permitido para el parámetro.

UsageProfileDefinition estructura

Describe un perfil de AWS Glue uso.

Campos

- `Name`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre del perfil de uso.
- `Description`: cadena de descripción de un máximo de 2048 bytes de largo, que coincide con el URI address multi-line string pattern.

  Una descripción del perfil de uso.
- `CreatedOn`: marca temporal.

  La fecha y la hora en que se creó el perfil de uso.
- `LastModifiedOn`: marca temporal.

  La fecha y la hora en que se modificó el perfil de uso por última vez.

Operaciones

- CreateUsageProfile acción (Python: create_usage_profile)
- GetUsageProfile acción (Python: get_usage_profile)
- UpdateUsageProfile acción (Python: update_usage_profile)
- DeleteUsageProfile acción (Python: delete_usage_profile)
- ListUsageProfiles acción (Python: list_usage_profiles)

CreateUsageProfile acción (Python: create_usage_profile)

Crea un perfil de uso AWS Glue .

Solicitud

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre del perfil de uso.
- `Description`: cadena de descripción de un máximo de 2048 bytes de largo, que coincide con el URI address multi-line string pattern.

Una descripción del perfil de uso.

- `Configuration`: obligatorio: objeto ProfileConfiguration.

  Un `ProfileConfiguration` objeto que especifica los valores de trabajo y sesión del perfil.

- `Tags`: matriz de mapas de pares de clave-valor, con 50 pares como máximo.

  Cada clave es una cadena UTF-8 con una longitud de entre 1 y 128 bytes.

  Cada valor es una cadena UTF-8 que no tiene más de 256 bytes de largo.

  Una lista de etiquetas aplicadas al perfil de uso.

Respuesta

- `Name`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre del perfil de uso que se creó.

Errores

- `InvalidInputException`
- `InternalServiceException`
- `AlreadyExistsException`
- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`
- `OperationNotSupportedException`

GetUsageProfile acción (Python: get_usage_profile)

Recupera información sobre el perfil de uso especificado. AWS Glue

Solicitud

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

El nombre del perfil de uso que se va a recuperar.

Respuesta

- `Name`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre del perfil de uso.

- `Description`: cadena de descripción de un máximo de 2048 bytes de largo, que coincide con el URI address multi-line string pattern.

  Una descripción del perfil de uso.

- `Configuration`: un objeto ProfileConfiguration.

  Un `ProfileConfiguration` objeto que especifica los valores de trabajo y sesión del perfil.

- `CreatedOn`: marca temporal.

  La fecha y la hora en que se creó el perfil de uso.

- `LastModifiedOn`: marca temporal.

  La fecha y la hora en que se modificó el perfil de uso por última vez.

Errores

- `InvalidInputException`
- `InternalServiceException`
- `EntityNotFoundException`
- `OperationTimeoutException`
- `OperationNotSupportedException`

UpdateUsageProfile acción (Python: update_usage_profile)

Actualiza un perfil de uso. AWS Glue

Solicitud

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre del perfil de uso.

- `Description`: cadena de descripción de un máximo de 2048 bytes de largo, que coincide con el URI address multi-line string pattern.

  Una descripción del perfil de uso.

- `Configuration`: obligatorio: objeto ProfileConfiguration.

  Un `ProfileConfiguration` objeto que especifica los valores de trabajo y sesión del perfil.

Respuesta

- `Name`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre del perfil de uso que se actualizó.

Errores

- `InvalidInputException`
- `InternalServiceException`
- `EntityNotFoundException`
- `OperationTimeoutException`
- `OperationNotSupportedException`
- `ConcurrentModificationException`

DeleteUsageProfile acción (Python: delete_usage_profile)

Elimina el perfil de uso especificado. AWS Glue

Solicitud

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre del perfil de uso que se va a eliminar.

Respuesta

- Sin parámetros de respuesta.

Errores

- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `OperationNotSupportedException`

ListUsageProfiles acción (Python: list_usage_profiles)

Enumere todos los perfiles de uso. AWS Glue

Solicitud

- `NextToken`: cadena UTF-8 de 400 000 bytes de largo como máximo.

  Token de continuación, incluido si se trata de una llamada de continuidad.
- `MaxResults`: número (entero), mayor que 1 y menor que 200.

  El número máximo de perfiles de uso que se devolverán en una sola respuesta.

Respuesta

- `Profiles`: matriz de objetos UsageProfileDefinition.

  Una lista de objetos del perfil de uso (`UsageProfileDefinition`).
- `NextToken`: cadena UTF-8 de 400 000 bytes de largo como máximo.

Un token de continuación, presente si el segmento de lista actual no es el último.

Errores

- `InternalServiceException`
- `OperationTimeoutException`
- `InvalidInputException`
- `OperationNotSupportedException`

API de machine learning

La API de machine learning describe los tipos de datos de machine learning e incluye la API para crear, eliminar o actualizar una transformación, o para iniciar la ejecución una tarea de machine learning.

Tipos de datos

- Estructura TransformParameters
- Estructura EvaluationMetrics
- Estructura MLTransform
- Estructura FindMatchesParameters
- Estructura FindMatchesMetrics
- Estructura ConfusionMatrix
- Estructura GlueTable
- Estructura TaskRun
- Estructura TransformFilterCriteria
- Estructura TransformSortCriteria
- Estructura TaskRunFilterCriteria
- Estructura TaskRunSortCriteria
- Estructura TaskRunProperties
- Estructura FindMatchesTaskRunProperties

- Estructura ImportLabelsTaskRunProperties
- Estructura ExportLabelsTaskRunProperties
- Estructura LabelingSetGenerationTaskRunProperties
- Estructura SchemaColumn
- Estructura TransformEncryption
- Estructura MLUserDataEncryption
- Estructura ColumnImportance

Estructura TransformParameters

Los parámetros específicos del algoritmo que están asociados a la transformación de machine learning.

Campos

- `TransformType` – Obligatorio: cadena UTF-8 (valores válidos: `FIND_MATCHES`).

  El tipo de transformación de machine learning.

  Para obtener más información sobre los tipos de transformaciones de machine learning, consulte Creación de transformaciones de machine learning.
- `FindMatchesParameters`: un objeto FindMatchesParameters.

  Los parámetros para el algoritmo de búsqueda de coincidencias.

Estructura EvaluationMetrics

Las métricas de evaluación proporcionan una estimación de la calidad de su transformación de machine learning.

Campos

- `TransformType` – Obligatorio: cadena UTF-8 (valores válidos: `FIND_MATCHES`).

  El tipo de transformación de machine learning.
- `FindMatchesMetrics`: un objeto FindMatchesMetrics.

Las métricas de evaluación para el algoritmo de búsqueda de coincidencias.

Estructura MLTransform

Una estructura para una transformación de machine learning.

Campos

- `TransformId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID de transformación único que se genera para la transformación de machine learning. Se garantiza que el ID es único y que no cambia.
- `Name`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Un nombre definido por el usuario para la transformación de machine learning. No se garantiza que los nombres sean únicos y se puedan cambiar en cualquier momento.
- `Description`: cadena de descripción de un máximo de 2048 bytes de largo, que coincide con el URI address multi-line string pattern.

  Un texto de descripción largo definido por el usuario para la transformación de machine learning. No se garantiza que las descripciones sean únicas y se puedan cambiar en cualquier momento.
- `Status`: cadena UTF-8 (valores válidos: `NOT_READY` | `READY` | `DELETING`).

  El estado actual de la transformación de machine learning.
- `CreatedOn`: marca temporal.

  Una marca temporal. La fecha y hora en las que se creó esta transformación de machine learning.
- `LastModifiedOn`: marca temporal.

  Una marca temporal. El último momento en el que se modificó esta transformación de machine learning.
- `InputRecordTables`: matriz de objetos GlueTable, con 10 estructuras como máximo.

  Una lista de definiciones de tabla de AWS Glue utilizadas por la transformación.
- `Parameters`: un objeto TransformParameters.

Un objeto `TransformParameters`. Puede utilizar parámetros para ajustar (personalizar) el comportamiento de la transformación de machine learning mediante la especificación de los datos que aprende y sus preferencias en diversas compensaciones (como precisión frente a exhaustividad o exactitud frente a costo).

- `EvaluationMetrics`: un objeto EvaluationMetrics.

  Un objeto `EvaluationMetrics`. Las métricas de evaluación proporcionan una estimación de la calidad de su transformación de machine learning.

- `LabelCount`: número (entero).

  Un identificador de recuentos para los archivos de etiquetado generados por AWS Glue para esta transformación. A medida que se crea una mejor transformación, puede descargar, etiquetar y cargar de forma iterativa el archivo de etiquetado.

- `Schema`: matriz de objetos SchemaColumn, con 100 estructuras como máximo.

  Una asignación de pares clave-valor que representa las columnas y tipos de datos con la que se puede ejecutar esta transformación. Tiene un límite superior de 100 columnas.

- `Role`: cadena UTF-8.

  El nombre o nombre de recurso de Amazon (ARN) del rol de IAM con los permisos obligatorios. Los permisos necesarios incluyen tanto permisos de rol de servicio de AWS Glue para recursos de AWS Glue, como permisos de Amazon S3 requeridos por la transformación.

  - Este rol necesita permisos de rol de servicio de AWS Glue para permitir el acceso a los recursos de AWS Glue. Consulte Asociar una política a usuarios de IAM que obtienen acceso a AWS Glue.
  - Este rol necesita permiso para los orígenes, los destinos, los directorios temporales, los scripts y las bibliotecas de Amazon Simple Storage Service (Amazon S3) utilizados por la ejecución de tareas para esta plataforma.

- `GlueVersion`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Custom string pattern #20.

  Este valor determina con qué versión de AWS Glue es compatible esta transformación de machine learning. Se recomienda Glue 1.0 para la mayoría de los clientes. Si el valor no está establecido, la compatibilidad de Glue se establece de forma predeterminada en Glue 0.9. Para obtener más información, consulte las Versiones de AWS Glue en la guía para desarrolladores.

- `MaxCapacity`: número (doble).

El número de unidades de procesamiento de datos (DPU) de AWS Glue que se asigna a ejecuciones de tareas para esta transformación. Puede asignar de 2 a 100 DPU; el valor predeterminado es 10. Una DPU es una medida relativa de la potencia de procesamiento que consta de 4 vCPU de capacidad de cómputo y 16 GB de memoria. Para obtener más información, consulte la página de precios de AWS Glue.

`MaxCapacity` es una opción mutuamente excluyente con `NumberOfWorkers` y `WorkerType`.

- Si se establecen `NumberOfWorkers` o `WorkerType`, no se puede establecer `MaxCapacity`.
- Si `MaxCapacity` se establece, no se pueden establecer `NumberOfWorkers` ni `WorkerType`.
- Si `WorkerType` se establece, `NumberOfWorkers` es obligatorio (y viceversa).
- `MaxCapacity` y `NumberOfWorkers` deben ser al menos 1.

Cuando el campo `WorkerType` se establece en un valor distinto a `Standard`, el campo `MaxCapacity` se establece automáticamente y se convierte a solo lectura.

- `WorkerType`: cadena UTF-8 (valores válidos: `Standard=""` | `G.1X=""` | `G.2X=""` | `G.025X=""` | `G.4X=""` | `G.8X=""` | `Z.2X=""`).

  El tipo de proceso de trabajo predefinido que se asigna cuando se ejecuta una tarea de esta transformación. Admite un valor de Standard, G.1X o G.2X.

  - Para el tipo de proceso de trabajo `Standard`, cada proceso de trabajo proporciona 4 vCPU, 16 GB de memoria y un disco de 50 GB y 2 ejecutores por trabajador.
  - Para el tipo de proceso de trabajo `G.1X`, cada proceso de trabajo proporciona 4 vCPU, 16 GB de memoria y un disco de 64 GB y 1 ejecutor por proceso de trabajo.
  - Para el tipo de proceso de trabajo `G.2X`, cada proceso de trabajo proporciona 8 vCPU, 32 GB de memoria y un disco de 128 GB y 1 ejecutor por proceso de trabajo.

  `MaxCapacity` es una opción mutuamente excluyente con `NumberOfWorkers` y `WorkerType`.

  - Si se establecen `NumberOfWorkers` o `WorkerType`, no se puede establecer `MaxCapacity`.
  - Si `MaxCapacity` se establece, no se pueden establecer `NumberOfWorkers` ni `WorkerType`.
  - Si `WorkerType` se establece, `NumberOfWorkers` es obligatorio (y viceversa).
  - `MaxCapacity` y `NumberOfWorkers` deben ser al menos 1.
- `NumberOfWorkers`: número (entero).

  El número de procesos de trabajo de un `workerType` definido que se asigna cuando se ejecuta una tarea de la transformación.

Si `WorkerType` se establece, `NumberOfWorkers` es obligatorio (y viceversa).

- `Timeout`: número (entero), como mínimo 1.

  El tiempo de espera en minutos de la transformación de machine learning.

- `MaxRetries`: número (entero).

  El número máximo de reintentos después de que se produzca un error de `MLTaskRun` de la transformación de machine learning.

- `TransformEncryption`: un objeto TransformEncryption.

  La configuración de cifrado en reposo de la transformación que se aplica al acceso a los datos del usuario. Las transformaciones de machine learning pueden acceder a los datos de usuario cifrados en Amazon S3 mediante KMS.

Estructura FindMatchesParameters

Los parámetros para configurar la transformación de búsquedas de coincidencias.

Campos

- `PrimaryKeyColumnName`: cadena UTF-8, con 1 byte como mínimo o más de 1024 bytes de largo, que coincide con el Single-line string pattern.

  El nombre de una columna que identifica de forma única las filas en la tabla de origen. Se utiliza para ayudar a identificar los registros coincidentes.

- `PrecisionRecallTradeoff`: número (doble), 1,0 como máximo.

  El valor seleccionado al ajustar su transformación para un equilibrio entre la precisión y la exhaustividad. Un valor de 0,5 implica que no hay preferencia; un valor de 1,0 significa un sesgo únicamente para precisión, y un valor de 0,0 implica un sesgo de exhaustividad. Puesto que se trata de una compensación, elegir valores próximos a 1,0 implica una exhaustividad menor y elegir resultados próximos a 0,0 origina una precisión muy baja.

  La métrica de precisión indica la frecuencia con la que el modelo es correcto cuando predice una coincidencia.

  La métrica de exhaustividad indica la frecuencia con la que el modelo predice la coincidencia en la coincidencia real.

- `AccuracyCostTradeoff`: número (doble), 1,0 como máximo.

  El valor seleccionado al ajustar su transformación para un equilibrio entre la exactitud y el costo. Un valor de 0,5 significa que el sistema equilibra los problemas de exactitud y costo. Un valor de 1,0 implica un sesgo únicamente para la exactitud, que normalmente da como resultado un costo considerablemente mayor, a veces bastante superior. Un valor de 0,0 hace referencia a un sesgo solo para el costo, lo que se traduce en una transformación `FindMatches` menos precisa, a veces con una exactitud inaceptable.

  La exactitud mide la facilidad con la que la transformación encuentra verdaderos positivos y verdaderos negativos. El aumento de la exactitud requiere más recursos informáticos y costos. Sin embargo, también genera una mayor exhaustividad.

  El costo mide la cantidad de recursos informáticos y, por lo tanto, dinero, necesario para ejecutar la transformación.

- `EnforceProvidedLabels`: booleano.

  El valor para activar o desactivar la opción para forzar la salida para que coincida con las etiquetas proporcionadas de los usuarios. Si el valor es `True`, la transformación `find matches` fuerza la salida para que coincida con las etiquetas proporcionadas. Los resultados anulan los resultados de combinación normales. Si el valor es `False`, la transformación `find matches` no que se respeten todas las etiquetas proporcionadas, y los resultados se basan en el modelo entrenado.

  Tenga en cuenta que si se establece este valor en true, el tiempo de ejecución de la combinación puede aumentar.

Estructura FindMatchesMetrics

Las métricas de evaluación para el algoritmo de búsqueda de coincidencias. La calidad de su transformación de machine learning se mide mediante su transformación para predecir algunas coincidencias y comparar los resultados con las coincidencias conocidas desde el mismo conjunto de datos. Las métricas de calidad se basan en un subconjunto de los datos, por lo que no son precisas.

Campos

- `AreaUnderPRCurve`: número (doble), 1,0 como máximo.

  El área bajo la curva de precisión/exhaustividad (AUPRC) es un número único que mide la calidad general de la transformación, que es independiente de la elección realizada para precisión en lugar

de la exhaustividad. Los valores más altos indican que cuenta con una compensación de precisión frente a exhaustividad más atractiva.

Para obtener más información, consulte Precisión y exhaustividad en Wikipedia.

- `Precision`: número (doble), 1,0 como máximo.

  La métrica de precisión indica la frecuencia con la que la transformación es correcta cuando predice una coincidencia. En concreto, mide la facilidad con la que de la transformación encuentra verdaderos positivos en el número total de verdaderos positivos posible.

  Para obtener más información, consulte Precisión y exhaustividad en Wikipedia.

- `Recall`: número (doble), 1,0 como máximo.

  La métrica de exhaustividad indica la frecuencia con la que la transformación predice la coincidencia en la coincidencia real. En concreto, mide la facilidad con la que de la transformación encuentra verdaderos positivos en los registros totales en los datos de origen.

  Para obtener más información, consulte Precisión y exhaustividad en Wikipedia.

- `F1`: número (doble), 1,0 como máximo.

  El valor-F máximo indica la exactitud de la transformación entre 0 y 1, donde 1 es la mejor exactitud.

  Para obtener más información, consulte Valor-F en Wikipedia.

- `ConfusionMatrix`: un objeto ConfusionMatrix.

  La matriz de confusión le muestra lo que la transformación está prediciendo de forma precisa y qué tipos de errores está realizando.

  Para obtener más información, consulte Matriz de confusión en Wikipedia.

- `ColumnImportances`: matriz de objetos ColumnImportance, con 100 estructuras como máximo.

  Una lista de estructuras de `ColumnImportance` que contienen métricas de importancia de columna, clasificadas en orden de importancia descendente.

Estructura ConfusionMatrix

La matriz de confusión le muestra lo que la transformación está prediciendo de forma precisa y qué tipos de errores está realizando.

Para obtener más información, consulte Matriz de confusión en Wikipedia.

Campos

- `NumTruePositives`: número (largo).

  El número de coincidencias en los datos que ha detectado la transformación correctamente, en la matriz de confusión para la transformación.

- `NumFalsePositives`: número (largo).

  El número de no coincidencias en los datos que la transformación ha clasificado incorrectamente como una coincidencia en la matriz de confusión para la transformación.

- `NumTrueNegatives`: número (largo).

  El número de no coincidencias en los datos que ha rechazado la transformación correctamente en la matriz de confusión para la transformación.

- `NumFalseNegatives`: número (largo).

  El número de coincidencias en los datos que no ha detectado la transformación en la matriz de confusión para la transformación.

Estructura GlueTable

La base de datos y la tabla en el AWS Glue Data Catalog que se usa para los datos de entrada y salida.

Campos

- `DatabaseName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Un nombre de base de datos en el AWS Glue Data Catalog.

- `TableName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Un nombre de tabla en el AWS Glue Data Catalog.

- `CatalogId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

Un identificador único para el AWS Glue Data Catalog.

- `ConnectionName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la conexión a AWS Glue Data Catalog.

- `AdditionalOptions`: una matriz de asignación de pares de clave-valor, con 1 par como mínimo y 10 como máximo.

  Cada clave es una cadena UTF-8 con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Cada valor es una cadena Description (Descripción), con 2048 bytes de largo como máximo, que coincide con URI address multi-line string pattern.

  Opciones adicionales para la tabla. Actualmente se admiten dos claves:

  - `pushDownPredicate`: filtra particiones sin tener que enumerar y leer todos los archivos del conjunto de datos.
  - `catalogPartitionPredicate`: para utilizar la eliminación de particiones del lado del servidor mediante los índices de particiones de AWS Glue Data Catalog.

Estructura TaskRun

Los parámetros de muestreo que están asociados a la transformación de machine learning.

Campos

- `TransformId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El identificador único de la transformación.

- `TaskRunId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El identificador único para esta ejecución de tareas.

- `Status`: cadena UTF-8 (valores válidos: `STARTING` | `RUNNING` | `STOPPING` | `STOPPED` | `SUCCEEDED` | `FAILED` | `TIMEOUT`).

El estado actual de la ejecución de tareas solicitada.

- `LogGroupName`: cadena UTF-8.

  Los nombres del grupo de registros para el registro segur asociados a esta ejecución de tareas.

- `Properties`: un objeto TaskRunProperties.

  Especifica las propiedades de configuración asociadas a esta ejecución de tareas.

- `ErrorString`: cadena UTF-8.

  La lista de cadenas de error asociadas a esta ejecución de tareas.

- `StartedOn`: marca temporal.

  La fecha y la hora en la que se inició esta tarea.

- `LastModifiedOn`: marca temporal.

  La última vez que se actualizó la ejecución de tareas solicitada.

- `CompletedOn`: marca temporal.

  La última vez que se completó la tarea solicitada.

- `ExecutionTime`: número (entero).

  El período (en segundos) que la ejecución de tareas consumió recursos.

Estructura TransformFilterCriteria

Los criterios utilizados para filtrar las transformaciones de machine learning.

Campos

- `Name`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Un nombre de transformación único que se utiliza para filtrar las transformaciones de machine learning.

- `TransformType`: cadena UTF-8 (valores válidos: `FIND_MATCHES`).

  El tipo de transformación de machine learning que se utiliza para filtrar las transformaciones de machine learning.

- `Status`: cadena UTF-8 (valores válidos: `NOT_READY` | `READY` | `DELETING`).

  Permite filtrar la lista de transformaciones de machine learning por el último estado conocido de las transformaciones (para indicar si una transformación se puede utilizar o no). Una de "NOT_READY", "READY" o "DELETING".

- `GlueVersion`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Custom string pattern #20.

  Este valor determina con qué versión de AWS Glue es compatible esta transformación de machine learning. Se recomienda Glue 1.0 para la mayoría de los clientes. Si el valor no está establecido, la compatibilidad de Glue se establece de forma predeterminada en Glue 0.9. Para obtener más información, consulte las Versiones de AWS Glue en la guía para desarrolladores.

- `CreatedBefore`: marca temporal.

  La fecha y hora antes de las que se crearon las transformaciones.

- `CreatedAfter`: marca temporal.

  La fecha y hora después de las que se crearon las transformaciones.

- `LastModifiedBefore`: marca temporal.

  Permite filtrar por las últimas transformaciones modificadas antes de esta fecha.

- `LastModifiedAfter`: marca temporal.

  Permite filtrar por las últimas transformaciones modificadas después de esta fecha.

- `Schema`: matriz de objetos SchemaColumn, con 100 estructuras como máximo.

  Permite filtrar por conjuntos de datos con un esquema específico. El objeto `Map<Column, Type>` es una matriz de pares clave-valor que representa el esquema que acepta esta transformación, donde `Column` es el nombre de una columna, y `Type` es el tipo de datos como, por ejemplo, un número entero o una cadena. Tiene un límite superior de 100 columnas.

Estructura TransformSortCriteria

Los criterios de ordenación que están asociados a la transformación de machine learning.

Campos

- `Column` – Obligatorio: cadena de UTF-8 (valores válidos: `NAME` | `TRANSFORM_TYPE` | `STATUS` | `CREATED` | `LAST_MODIFIED`).

  La columna que se va a utilizar en los criterios de ordenación que están asociados a la transformación de machine learning.
- `SortDirection`: obligatorio: cadena UTF-8 (valores válidos: `DESCENDING` | `ASCENDING`).

  La dirección de ordenación que se va a utilizar en los criterios de ordenación que están asociados a la transformación de machine learning.

Estructura TaskRunFilterCriteria

Los criterios que se utilizan para filtrar las ejecuciones de tareas durante la transformación de machine learning.

Campos

- `TaskRunType`: cadena UTF-8 (valores válidos: `EVALUATION` | `LABELING_SET_GENERATION` | `IMPORT_LABELS` | `EXPORT_LABELS` | `FIND_MATCHES`).

  El tipo de ejecución de tareas.
- `Status`: cadena UTF-8 (valores válidos: `STARTING` | `RUNNING` | `STOPPING` | `STOPPED` | `SUCCEEDED` | `FAILED` | `TIMEOUT`).

  El estado actual de la ejecución de tareas.
- `StartedBefore`: marca temporal.

  Permite filtrar por ejecuciones de tareas iniciadas antes de esa fecha.
- `StartedAfter`: marca temporal.

  Permite filtrar por ejecuciones de tareas iniciadas después de esa fecha.

Estructura TaskRunSortCriteria

Los criterios de ordenación que se utilizan para filtrar la lista de ejecuciones de tareas durante la transformación de machine learning.

Campos

- `Column` – Obligatorio: cadena UTF-8 (valores válidos: `TASK_RUN_TYPE` | `STATUS` | `STARTED`).

  La columna que se va a utilizar para ordenar la lista de ejecuciones de tareas durante la transformación de machine learning.

- `SortDirection`: obligatorio: cadena UTF-8 (valores válidos: `DESCENDING` | `ASCENDING`).

  La dirección de ordenación que se va a utilizar para ordenar la lista de ejecuciones de tareas durante la transformación de machine learning.

Estructura TaskRunProperties

Las propiedades de configuración de la ejecución de tareas.

Campos

- `TaskType`: cadena UTF-8 (valores válidos: `EVALUATION` | `LABELING_SET_GENERATION` | `IMPORT_LABELS` | `EXPORT_LABELS` | `FIND_MATCHES`).

  El tipo de ejecución de tareas.

- `ImportLabelsTaskRunProperties`: un objeto ImportLabelsTaskRunProperties.

  Las propiedades de configuración de una ejecución de tareas de etiquetas de importación.

- `ExportLabelsTaskRunProperties`: un objeto ExportLabelsTaskRunProperties.

  Las propiedades de configuración de una ejecución de tareas de etiquetas de exportación.

- `LabelingSetGenerationTaskRunProperties`: un objeto LabelingSetGenerationTaskRunProperties.

  Las propiedades de configuración de una ejecución de tareas de generación de conjuntos de etiquetas.

- `FindMatchesTaskRunProperties`: un objeto FindMatchesTaskRunProperties.

  Las propiedades de configuración para la ejecución de tareas de búsqueda de coincidencias.

Estructura FindMatchesTaskRunProperties

Especifica las propiedades de configuración para la ejecución de tareas de búsqueda de coincidencias.

Campos

- `JobId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID de trabajo para la ejecución de tareas de búsqueda de coincidencias.
- `JobName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre asignado al trabajo para la ejecución de tareas de búsqueda de coincidencias.
- `JobRunId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID de ejecución de tareas para la ejecución de tareas de búsqueda de coincidencias.

Estructura ImportLabelsTaskRunProperties

Especifica las propiedades de configuración de una ejecución de tareas de etiquetas de importación.

Campos

- `InputS3Path`: cadena UTF-8.

  La ruta de Amazon Simple Storage Service (Amazon S3) desde la que importará las etiquetas.
- `Replace`: booleano.

  Indica si se van a sobrescribir las etiquetas existentes.

Estructura ExportLabelsTaskRunProperties

Especifica las propiedades de configuración de una ejecución de tareas de etiquetas de exportación.

Campos

- `OutputS3Path`: cadena UTF-8.

La ruta de Amazon Simple Storage Service (Amazon S3) en la que exportará las etiquetas.

Estructura LabelingSetGenerationTaskRunProperties

Especifica las propiedades de configuración de una ejecución de tareas de generación de conjuntos de etiquetas.

Campos

- `OutputS3Path`: cadena UTF-8.

  La ruta de Amazon Simple Storage Service (Amazon S3) en la que generará el conjunto de etiquetas.

Estructura SchemaColumn

Un par clave-valor que representa una columna y tipo de datos con el que se puede ejecutar esta transformación. El parámetro `Schema` de `MLTransform` puede contener hasta 100 de estas estructuras.

Campos

- `Name`: cadena UTF-8, con 1 byte como mínimo o más de 1024 bytes de largo, que coincide con el Single-line string pattern.

  El nombre de la columna.

- `DataType`: cadena UTF-8 con un máximo de 131072 bytes de largo, que coincide con el Single-line string pattern.

  El tipo de datos en la columna.

Estructura TransformEncryption

La configuración de cifrado en reposo de la transformación que se aplica al acceso a los datos del usuario. Las transformaciones de machine learning pueden acceder a los datos de usuario cifrados en Amazon S3 mediante KMS.

Además, las etiquetas importadas y las transformaciones entrenadas se pueden cifrar mediante una clave KMS proporcionada por el cliente.

Campos

- `MlUserDataEncryption`: un objeto MLUserDataEncryption.

  Objeto `MLUserDataEncryption` que contiene el modo de cifrado y el ID de clave KMS proporcionado por el cliente.

- `TaskRunSecurityConfigurationName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la configuración de seguridad.

Estructura MLUserDataEncryption

La configuración de cifrado en reposo de la transformación que se aplica al acceso a los datos del usuario.

Campos

- `MlUserDataEncryptionMode`: obligatorio: cadena UTF-8 (valores válidos: `DISABLED` | `SSE-KMS="SSEKMS"`).

  Modo de cifrado aplicado a los datos del usuario. Los valores válidos son:

  - DISABLED: el cifrado está desactivado
  - SSEKMS: uso del cifrado del lado del servidor con AWS Key Management Service (SSE-KMS) para los datos de usuario almacenados en Amazon S3.

- `KmsKeyId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  ID de la clave KMS proporcionada por el cliente.

Estructura ColumnImportance

Estructura que contiene el nombre de columna y la puntuación de importancia de columna para una columna.

La importancia de columnas ayuda a comprender cómo contribuyen las columnas al modelo, al identificar qué columnas en sus registros son más importantes que otras.

Campos

- `ColumnName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de una columna.
- `Importance`: número (doble), 1,0 como máximo.

  Puntuación de importancia de columna para la columna, como decimal.

Operaciones

- Acción CreateMLTransform (Python: create_ml_transform)
- Acción UpdateMLTransform (Python: update_ml_transform)
- Acción DeleteMLTransform (Python: delete_ml_transform)
- Acción GetMLTransform (Python: get_ml_transform)
- Acción GetMLTransforms (Python: get_ml_transforms)
- Acción ListMLTransforms (Python: list_ml_transforms)
- Acción StartMLEvaluationTaskRun (Python: start_ml_evaluation_task_run)
- Acción StartMLLabelingSetGenerationTaskRun (Python: start_ml_labeling_set_generation_task_run)
- Acción GetMLTaskRun (Python: get_ml_task_run)
- Acción GetMLTaskRuns (Python: get_ml_task_runs)
- Acción CancelMLTaskRun (Python: cancel_ml_task_run)
- Acción StartExportLabelsTaskRun (Python: start_export_labels_task_run)
- Acción StartImportLabelsTaskRun (Python: start_import_labels_task_run)

Acción CreateMLTransform (Python: create_ml_transform)

Permite crear una transformación de machine learning de AWS Glue. En esta operación se crea la transformación y todos los parámetros necesarios para su entrenamiento.

Realice esta operación como el primer paso del proceso de uso de una transformación de machine learning (como la transformación `FindMatches`) para la deduplicación de datos. Puede proporcionar una `Description` opcional, además de los parámetros que desee utilizar para su algoritmo.

Además, debe especificar determinados parámetros para las tareas que AWS Glue ejecuta en su nombre como parte del aprendizaje a partir de los datos y la creación de una transformación de machine learning de alta calidad. Estos parámetros incluyen `Role` y, de forma opcional,, `AllocatedCapacity`, `Timeout` y `MaxRetries`. Para obtener más información, consulte Trabajos.

Solicitud

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre único que asigna a la transformación cuando la creó.

- `Description`: cadena de descripción de un máximo de 2048 bytes de largo, que coincide con el URI address multi-line string pattern.

  Una descripción de la transformación de machine learning que se está definiendo. El valor predeterminado es una cadena vacía.

- `InputRecordTables` – Obligatorio: matriz de objetos GlueTable, con 10 estructuras como máximo.

  Una lista de definiciones de tabla de AWS Glue utilizadas por la transformación.

- `Parameters`: obligatorio: objeto TransformParameters.

  Los parámetros de algoritmo específicos para el tipo de transformación utilizado. Dependen condicionalmente del tipo de transformación.

- `Role` – Obligatorio: cadena UTF-8.

  El nombre o nombre de recurso de Amazon (ARN) del rol de IAM con los permisos obligatorios. Los permisos necesarios incluyen tanto permisos de rol de servicio de AWS Glue para recursos de AWS Glue, como permisos de Amazon S3 requeridos por la transformación.

  - Este rol necesita permisos de rol de servicio de AWS Glue para permitir el acceso a los recursos de AWS Glue. Consulte Asociar una política a usuarios de IAM que obtienen acceso a AWS Glue.

  - Este rol necesita permiso para los orígenes, los destinos, los directorios temporales, los scripts y las bibliotecas de Amazon Simple Storage Service (Amazon S3) utilizados por la ejecución de tareas para esta plataforma.

- `GlueVersion`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Custom string pattern #20.

  Este valor determina con qué versión de AWS Glue es compatible esta transformación de machine learning. Se recomienda Glue 1.0 para la mayoría de los clientes. Si el valor no está establecido, la compatibilidad de Glue se establece de forma predeterminada en Glue 0.9. Para obtener más información, consulte las Versiones de AWS Glue en la guía para desarrolladores.

- `MaxCapacity`: número (doble).

  El número de unidades de procesamiento de datos (DPU) de AWS Glue que se asigna a ejecuciones de tareas para esta transformación. Puede asignar de 2 a 100 DPU; el valor predeterminado es 10. Una DPU es una medida relativa de la potencia de procesamiento que consta de 4 vCPU de capacidad de cómputo y 16 GB de memoria. Para obtener más información, consulte la página de precios de AWS Glue.

  `MaxCapacity` es una opción mutuamente excluyente con `NumberOfWorkers` y `WorkerType`.

  - Si se establecen `NumberOfWorkers` o `WorkerType`, no se puede establecer `MaxCapacity`.
  - Si `MaxCapacity` se establece, no se pueden establecer `NumberOfWorkers` ni `WorkerType`.
  - Si `WorkerType` se establece, `NumberOfWorkers` es obligatorio (y viceversa).
  - `MaxCapacity` y `NumberOfWorkers` deben ser al menos 1.

  Cuando el campo `WorkerType` se establece en un valor distinto a `Standard`, el campo `MaxCapacity` se establece automáticamente y se convierte a solo lectura.

  Cuando el campo `WorkerType` se establece en un valor distinto a `Standard`, el campo `MaxCapacity` se establece automáticamente y se convierte a solo lectura.

- `WorkerType`: cadena UTF-8 (valores válidos: `Standard=""` | `G.1X=""` | `G.2X=""` | `G.025X=""` | `G.4X=""` | `G.8X=""` | `Z.2X=""`).

  El tipo de proceso de trabajo predefinido que se asigna cuando se ejecuta esta tarea. Admite un valor de Standard, G.1X o G.2X.

  - Para el tipo de proceso de trabajo `Standard`, cada proceso de trabajo proporciona 4 vCPU, 16 GB de memoria y un disco de 50 GB y 2 ejecutores por trabajador.

  - Para el tipo de proceso de trabajo `G.1X`, cada proceso de trabajo proporciona 4 vCPU, 16 GB de memoria y un disco de 64 GB y 1 ejecutor por proceso de trabajo.
  - Para el tipo de proceso de trabajo `G.2X`, cada proceso de trabajo proporciona 8 vCPU, 32 GB de memoria y un disco de 128 GB y 1 ejecutor por proceso de trabajo.

  `MaxCapacity` es una opción mutuamente excluyente con `NumberOfWorkers` y `WorkerType`.

  - Si se establecen `NumberOfWorkers` o `WorkerType`, no se puede establecer `MaxCapacity`.
  - Si `MaxCapacity` se establece, no se pueden establecer `NumberOfWorkers` ni `WorkerType`.
  - Si `WorkerType` se establece, `NumberOfWorkers` es obligatorio (y viceversa).
  - `MaxCapacity` y `NumberOfWorkers` deben ser al menos 1.
- `NumberOfWorkers`: número (entero).

  El número de procesos de trabajo de `workerType` definido que se asignan cuando se ejecuta esta tarea.

  Si `WorkerType` se establece, `NumberOfWorkers` es obligatorio (y viceversa).
- `Timeout`: número (entero), como mínimo 1.

  El tiempo de espera de la ejecución de tareas para esta transformación en minutos. Es el tiempo máximo que una ejecución de flujo para esta transformación puede consumir recursos antes de que se termine y cambie al estado `TIMEOUT`. El valor predeterminado es 2880 minutos (48 horas).
- `MaxRetries`: número (entero).

  El número máximo de veces que se intenta una tarea para esta transformación después de que se produzca un error en una ejecución de tareas.
- `Tags`: matriz de mapas de pares de clave-valor, con 50 pares como máximo.

  Cada clave es una cadena UTF-8 con una longitud de entre 1 y 128 bytes.

  Cada valor es una cadena UTF-8 que no tiene más de 256 bytes de largo.

  Las etiquetas que se van a utilizar con esta transformación de machine learning. Puede utilizar etiquetas para limitar el acceso la transformación de machine learning. Para obtener más información acerca de las etiquetas en AWS Glue, consulte Etiquetas de AWS en AWS Glue en la guía para desarrolladores.
- `TransformEncryption`: un objeto TransformEncryption.

La configuración de cifrado en reposo de la transformación que se aplica al acceso a los datos del usuario. Las transformaciones de machine learning pueden acceder a los datos de usuario cifrados en Amazon S3 mediante KMS.

Respuesta

- `TransformId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Un identificador único que se genera para la transformación.

Errores

- `AlreadyExistsException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`
- `AccessDeniedException`
- `ResourceNumberLimitExceededException`
- `IdempotentParameterMismatchException`

Acción UpdateMLTransform (Python: update_ml_transform)

Permite actualizar una transformación de machine learning existente. Realice esta operación para ajustar los parámetros del algoritmo para conseguir mejores resultados.

Después de realizar esta operación, puede realizar la operación `StartMLEvaluationTaskRun` para evaluar la facilidad con la que los nuevos parámetros consiguieron sus objetivos (como la mejora de la calidad de la transformación de machine learning o la transformación en una solución más rentable).

Solicitud

- `TransformId`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

Un identificador único que se generó cuando se creó la transformación.

- `Name`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre único que asignó a la transformación cuando la creó.

- `Description`: cadena de descripción de un máximo de 2048 bytes de largo, que coincide con el URI address multi-line string pattern.

  Una descripción de la transformación. El valor predeterminado es una cadena vacía.

- `Parameters`: un objeto TransformParameters.

  Los parámetros de configuración específicos para el tipo de transformación (algoritmo) utilizado. Dependen condicionalmente del tipo de transformación.

- `Role`: cadena UTF-8.

  El nombre o nombre de recurso de Amazon (ARN) del rol de IAM con los permisos obligatorios.

- `GlueVersion`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Custom string pattern #20.

  Este valor determina con qué versión de AWS Glue es compatible esta transformación de machine learning. Se recomienda Glue 1.0 para la mayoría de los clientes. Si el valor no está establecido, la compatibilidad de Glue se establece de forma predeterminada en Glue 0.9. Para obtener más información, consulte las Versiones de AWS Glue en la guía para desarrolladores.

- `MaxCapacity`: número (doble).

  El número de unidades de procesamiento de datos (DPU) de AWS Glue que se asigna a ejecuciones de tareas para esta transformación. Puede asignar de 2 a 100 DPU; el valor predeterminado es 10. Una DPU es una medida relativa de la potencia de procesamiento que consta de 4 vCPU de capacidad de cómputo y 16 GB de memoria. Para obtener más información, consulte la página de precios de AWS Glue.

  Cuando el campo `WorkerType` se establece en un valor distinto a `Standard`, el campo `MaxCapacity` se establece automáticamente y se convierte a solo lectura.

- `WorkerType`: cadena UTF-8 (valores válidos: `Standard=""` | `G.1X=""` | `G.2X=""` | `G.025X=""` | `G.4X=""` | `G.8X=""` | `Z.2X=""`).

El tipo de proceso de trabajo predefinido que se asigna cuando se ejecuta esta tarea. Admite un valor de Standard, G.1X o G.2X.

- Para el tipo de proceso de trabajo `Standard`, cada proceso de trabajo proporciona 4 vCPU, 16 GB de memoria y un disco de 50 GB y 2 ejecutores por trabajador.
- Para el tipo de proceso de trabajo `G.1X`, cada proceso de trabajo proporciona 4 vCPU, 16 GB de memoria y un disco de 64 GB y 1 ejecutor por proceso de trabajo.
- Para el tipo de proceso de trabajo `G.2X`, cada proceso de trabajo proporciona 8 vCPU, 32 GB de memoria y un disco de 128 GB y 1 ejecutor por proceso de trabajo.

- `NumberOfWorkers`: número (entero).

  El número de procesos de trabajo de `workerType` definido que se asignan cuando se ejecuta esta tarea.

- `Timeout`: número (entero), como mínimo 1.

  El tiempo de espera de una ejecución de tareas para esta transformación en minutos. Es el tiempo máximo que una ejecución de flujo para esta transformación puede consumir recursos antes de que se termine y cambie al estado `TIMEOUT`. El valor predeterminado es 2880 minutos (48 horas).

- `MaxRetries`: número (entero).

  El número máximo de veces que se intenta una tarea para esta transformación después de que se produzca un error en una ejecución de tareas.

Respuesta

- `TransformId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El identificador único para la transformación que se ha actualizado.

Errores

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

- `AccessDeniedException`

Acción DeleteMLTransform (Python: delete_ml_transform)

Elimina una transformación de machine learning de AWS Glue. Las transformaciones de machine learning son un tipo especial de transformación que utiliza el machine learning para aprender los detalles de la transformación que se va a realizar a través del aprendizaje de ejemplos proporcionados por los humanos. Luego, AWS Glue guarda estas transformaciones. Si ya no necesita una transformación, puede eliminarla con `DeleteMLTransforms`. Sin embargo, los trabajos de AWS Glue que todavía hagan referencia a la transformación eliminada dejarán de ser correctos.

Solicitud

- `TransformId`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El identificador único de la transformación que eliminar.

Respuesta

- `TransformId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El identificador único de la transformación que se ha eliminado.

Errores

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

Acción GetMLTransform (Python: get_ml_transform)

Obtiene un artefacto de transformación de machine learning de AWS Glue y todos sus metadatos correspondientes. Las transformaciones de machine learning son un tipo especial de transformación

que utiliza el machine learning para aprender los detalles de la transformación que se va a realizar a través del aprendizaje de ejemplos proporcionados por los humanos. Luego, AWS Glue guarda estas transformaciones. Puede recuperar sus metadatos con `GetMLTransform`.

Solicitud

- `TransformId`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El identificador único de la transformación, generado en el momento en que se creó la transformación.

Respuesta

- `TransformId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El identificador único de la transformación, generado en el momento en que se creó la transformación.
- `Name`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre único proporcionado a la transformación en el momento en el que se creó.
- `Description`: cadena de descripción de un máximo de 2048 bytes de largo, que coincide con el URI address multi-line string pattern.

  Una descripción de la transformación.
- `Status`: cadena UTF-8 (valores válidos: `NOT_READY` | `READY` | `DELETING`).

  El último estado conocido de la transformación (para indicar si se puede utilizar o no). Una de "NOT_READY", "READY" o "DELETING".
- `CreatedOn`: marca temporal.

  La fecha y hora de cuando se creó la transformación.
- `LastModifiedOn`: marca temporal.

  La fecha y hora de cuando se modificó la transformación por última vez.
- `InputRecordTables`: matriz de objetos GlueTable, con 10 estructuras como máximo.

Una lista de definiciones de tabla de AWS Glue utilizadas por la transformación.

- `Parameters`: un objeto TransformParameters.

  Los parámetros de configuración específicos para el algoritmo utilizado.

- `EvaluationMetrics`: un objeto EvaluationMetrics.

  Las métricas de evaluación más recientes.

- `LabelCount`: número (entero).

  El número de etiquetas disponibles para esta transformación.

- `Schema`: matriz de objetos SchemaColumn, con 100 estructuras como máximo.

  El objeto `Map<Column, Type>` que representa al esquema que acepta esta transformación. Tiene un límite superior de 100 columnas.

- `Role`: cadena UTF-8.

  El nombre o nombre de recurso de Amazon (ARN) del rol de IAM con los permisos obligatorios.

- `GlueVersion`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Custom string pattern #20.

  Este valor determina con qué versión de AWS Glue es compatible esta transformación de machine learning. Se recomienda Glue 1.0 para la mayoría de los clientes. Si el valor no está establecido, la compatibilidad de Glue se establece de forma predeterminada en Glue 0.9. Para obtener más información, consulte las Versiones de AWS Glue en la guía para desarrolladores.

- `MaxCapacity`: número (doble).

  El número de unidades de procesamiento de datos (DPU) de AWS Glue que se asigna a ejecuciones de tareas para esta transformación. Puede asignar de 2 a 100 DPU; el valor predeterminado es 10. Una DPU es una medida relativa de la potencia de procesamiento que consta de 4 vCPU de capacidad de cómputo y 16 GB de memoria. Para obtener más información, consulte la página de precios de AWS Glue.

  Cuando el campo `WorkerType` se establece en un valor distinto a `Standard`, el campo `MaxCapacity` se establece automáticamente y se convierte a solo lectura.

- `WorkerType`: cadena UTF-8 (valores válidos: `Standard=""` | `G.1X=""` | `G.2X=""` | `G.025X=""` | `G.4X=""` | `G.8X=""` | `Z.2X=""`).

El tipo de proceso de trabajo predefinido que se asigna cuando se ejecuta esta tarea. Admite un valor de Standard, G.1X o G.2X.

- Para el tipo de proceso de trabajo `Standard`, cada proceso de trabajo proporciona 4 vCPU, 16 GB de memoria y un disco de 50 GB y 2 ejecutores por trabajador.
- Para el tipo de proceso de trabajo `G.1X`, cada proceso de trabajo proporciona 4 vCPU, 16 GB de memoria y un disco de 64 GB y 1 ejecutor por proceso de trabajo.
- Para el tipo de proceso de trabajo `G.2X`, cada proceso de trabajo proporciona 8 vCPU, 32 GB de memoria y un disco de 128 GB y 1 ejecutor por proceso de trabajo.

- `NumberOfWorkers`: número (entero).

  El número de procesos de trabajo de `workerType` definido que se asignan cuando se ejecuta esta tarea.

- `Timeout`: número (entero), como mínimo 1.

  El tiempo de espera de una ejecución de tareas para esta transformación en minutos. Es el tiempo máximo que una ejecución de flujo para esta transformación puede consumir recursos antes de que se termine y cambie al estado `TIMEOUT`. El valor predeterminado es 2880 minutos (48 horas).

- `MaxRetries`: número (entero).

  El número máximo de veces que se intenta una tarea para esta transformación después de que se produzca un error en una ejecución de tareas.

- `TransformEncryption`: un objeto TransformEncryption.

  La configuración de cifrado en reposo de la transformación que se aplica al acceso a los datos del usuario. Las transformaciones de machine learning pueden acceder a los datos de usuario cifrados en Amazon S3 mediante KMS.

Errores

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

Acción GetMLTransforms (Python: get_ml_transforms)

Permite obtener una lista ordenable y filtrable de transformaciones de machine learning de AWS Glue. Las transformaciones de machine learning son un tipo especial de transformación que utiliza el machine learning para aprender los detalles de la transformación que se va a realizar a través del aprendizaje de ejemplos proporcionados por los humanos. Luego, AWS Glue guarda estas transformaciones y el usuario puede recuperar metadatos con la llamada a `GetMLTransforms`.

Solicitud

- `NextToken`: cadena UTF-8.

  Un token paginado para compensar los resultados.
- `MaxResults`: número (entero) que no es inferior a 1 ni es superior a 1000.

  El número máximo de resultados que devolver.
- `Filter`: un objeto TransformFilterCriteria.

  Los criterios de transformación del filtro.
- `Sort`: un objeto TransformSortCriteria.

  Los criterios de ordenación.

Respuesta

- `Transforms` (obligatorio): una matriz de objetos MLTransform.

  Una lista de transformaciones de machine learning.
- `NextToken`: cadena UTF-8.

  Un token de paginación, si hay disponibles más resultados.

Errores

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

Acción ListMLTransforms (Python: list_ml_transforms)

Recupera una lista ordenable y filtrable de las transformaciones de machine learning de AWS Glue existentes en esta cuenta de AWS, o los recursos con la etiqueta especificada. Esta operación toma el campo `Tags` opcional, que se puede utilizar como filtro en las respuestas para que los recursos etiquetados se devuelvan agrupados. Si elige utilizar el filtrado de etiquetas, solo se recuperan los recursos con las etiquetas especificadas.

Solicitud

- `NextToken`: cadena UTF-8.

  Token de continuación, si se trata de una solicitud de continuidad.
- `MaxResults`: número (entero) que no es inferior a 1 ni es superior a 1000.

  Tamaño máximo de una lista que se devolverá.
- `Filter`: un objeto TransformFilterCriteria.

  Valor `TransformFilterCriteria` utilizado para filtrar las transformaciones de machine learning.
- `Sort`: un objeto TransformSortCriteria.

  Valor `TransformSortCriteria` utilizado para ordenar las transformaciones de machine learning.
- `Tags`: matriz de mapas de pares de clave-valor, con 50 pares como máximo.

  Cada clave es una cadena UTF-8 con una longitud de entre 1 y 128 bytes.

  Cada valor es una cadena UTF-8 que no tiene más de 256 bytes de largo.

  Especifica que se devuelvan solamente los recursos etiquetados.

Respuesta

- `TransformIds` – Obligatorio: una matriz de cadenas UTF-8.

  Los identificadores de todas las transformaciones de machine learning de la cuenta o las transformaciones de machine learning con las etiquetas especificadas.
- `NextToken`: cadena UTF-8.

Token de continuación, si la lista devuelta no contiene la última métrica disponible.

Errores

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

Acción StartMLEvaluationTaskRun (Python: start_ml_evaluation_task_run)

Permite iniciar una tarea para calcular la calidad de la transformación.

Al proporcionar conjuntos de etiquetas como ejemplos de verdad, machine learning de AWS Glue utiliza algunos ejemplos para aprender de ellos. El resto de etiquetas se utilizan como una prueba para estimar la calidad.

Permite devolver un identificador único para la ejecución. Puede llamar a `GetMLTaskRun` para obtener más información sobre las estadísticas de `EvaluationTaskRun`.

Solicitud

- `TransformId`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El identificador único de la transformación de machine learning.

Respuesta

- `TaskRunId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El identificador único asociado a esta ejecución.

Errores

- `EntityNotFoundException`

- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`
- `ConcurrentRunsExceededException`
- `MLTransformNotReadyException`

Acción StartMLLabelingSetGenerationTaskRun (Python: start_ml_labeling_set_generation_task_run)

Permite iniciar el flujo de trabajo de aprendizaje activo para su transformación de machine learning para mejorar la calidad de la transformación mediante la generación de conjuntos de etiquetas y la adición de etiquetas.

Cuando `StartMLLabelingSetGenerationTaskRun` finaliza, AWS Glue dispondrá de un “conjunto de etiquetas” generado o un conjunto de preguntas que los humanos tendrán que responder.

En el caso de la transformación `FindMatches`, estas preguntas tienen la siguiente forma: "¿Cuál es la forma correcta de agrupar estas filas en grupos compuestos enteramente por registros coincidentes?".

Una vez que el proceso de etiquetado finalice, puede cargar etiquetas con una llamada a `StartImportLabelsTaskRun`. Una vez que `StartImportLabelsTaskRun` acabe, todas las ejecuciones futuras de la transformación de machine learning utilizarán las etiquetas nuevas y mejoradas, y realizarán una transformación de alta calidad.

Solicitud

- `TransformId`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El identificador único de la transformación de machine learning.

- `OutputS3Path` – Obligatorio: cadena UTF-8.

  La ruta de Amazon Simple Storage Service (Amazon S3) en la que generará el conjunto de etiquetas.

Respuesta

- `TaskRunId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El identificador de ejecución único que está asociado a esta ejecución de tareas.

Errores

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`
- `ConcurrentRunsExceededException`

Acción GetMLTaskRun (Python: get_ml_task_run)

Permite obtener detalles para una ejecución de tareas específica en una transformación de machine learning. Las ejecuciones de tareas de machine learning son tareas asíncronas que AWS Glue ejecuta en nombre del usuario como parte de varios flujos de trabajo de machine learning. Puede comprobar las estadísticas de cualquier ejecución de tareas llamando a `GetMLTaskRun` con `TaskRunID` y `TransformID` de la transformación principal.

Solicitud

- `TransformId`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El identificador único de la transformación de machine learning.
- `TaskRunId`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El identificador único de la ejecución de tareas.

Respuesta

- `TransformId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El identificador único de la ejecución de tareas.

- `TaskRunId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El identificador de ejecución único asociado a esta ejecución.

- `Status`: cadena UTF-8 (valores válidos: `STARTING` | `RUNNING` | `STOPPING` | `STOPPED` | `SUCCEEDED` | `FAILED` | `TIMEOUT`).

  El estado de esta ejecución de tareas.

- `LogGroupName`: cadena UTF-8.

  Los nombres de los grupos de registros asociados a la ejecución de tareas.

- `Properties`: un objeto TaskRunProperties.

  La lista de propiedades asociadas a la ejecución de tareas.

- `ErrorString`: cadena UTF-8.

  Las cadenas de error asociadas a la ejecución de tareas.

- `StartedOn`: marca temporal.

  La fecha y la hora en las que se inició esta ejecución de tareas.

- `LastModifiedOn`: marca temporal.

  La fecha y hora de cuando se modificó esta ejecución de tareas por última vez.

- `CompletedOn`: marca temporal.

  La fecha y hora de cuando se completó esta ejecución de tareas.

- `ExecutionTime`: número (entero).

  El período (en segundos) que la ejecución de tareas consumió recursos.

Errores

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

Acción GetMLTaskRuns (Python: get_ml_task_runs)

Permite obtener una lista de ejecuciones para una transformación de machine learning. Las ejecuciones de tareas de machine learning son tareas asíncronas que AWS Glue ejecuta en nombre del usuario como parte de varios flujos de trabajo de machine learning. Puede obtener una lista ordenable y filtrable de ejecuciones de tareas de machine learning con `GetMLTaskRuns` con `TransformID` de la transformación principal y otros parámetros opcionales tal y como se indica en esta sección.

En esta operación se devuelve una lista de ejecuciones históricas y se debe paginar.

Solicitud

- `TransformId`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El identificador único de la transformación de machine learning.
- `NextToken`: cadena UTF-8.

  Un token para la paginación de los resultados. El valor predeterminado es vacío.
- `MaxResults`: número (entero) que no es inferior a 1 ni es superior a 1000.

  El número máximo de resultados que devolver.
- `Filter`: un objeto TaskRunFilterCriteria.

  Los criterios de filtro, en la estructura `TaskRunFilterCriteria`, para la ejecución de tareas.
- `Sort`: un objeto TaskRunSortCriteria.

  Los criterios de ordenación, en la estructura `TaskRunSortCriteria`, para la ejecución de tareas.

Respuesta

- `TaskRuns`: matriz de objetos TaskRun.

  La lista de ejecuciones de tareas que se asocian a la transformación.

- `NextToken`: cadena UTF-8.

  Un token de paginación, si hay disponibles más resultados.

Errores

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

Acción CancelMLTaskRun (Python: cancel_ml_task_run)

Permite cancelar (detener) una ejecución de tareas. Las ejecuciones de tareas de machine learning son tareas asíncronas que AWS Glue ejecuta en nombre del usuario como parte de varios flujos de trabajo de machine learning. Puede cancelar una ejecución de tareas de machine learning en cualquier momento llamando a `CancelMLTaskRun` con `TransformID` de la transformación principal de una ejecución de tareas y `TaskRunId` de la ejecución de tareas.

Solicitud

- `TransformId`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El identificador único de la transformación de machine learning.

- `TaskRunId`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Un identificador único de la ejecución de tareas.

Respuesta

- `TransformId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El identificador único de la transformación de machine learning.

- `TaskRunId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El identificador único de la ejecución de tareas.

- `Status`: cadena UTF-8 (valores válidos: `STARTING` | `RUNNING` | `STOPPING` | `STOPPED` | `SUCCEEDED` | `FAILED` | `TIMEOUT`).

  El estado de esta ejecución.

Errores

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

Acción StartExportLabelsTaskRun (Python: start_export_labels_task_run)

Permite iniciar una tarea asíncrona para exportar todos los datos etiquetados para una transformación determinada. Esta tarea es la única llamada de la API relacionada con la etiqueta que no forma parte del flujo de trabajo de aprendizaje activo normal. Normalmente, utiliza `StartExportLabelsTaskRun` cuando desea trabajar con todas las etiquetas existentes a la vez, como cuando desea quitar o cambiar etiquetas que se enviaron anteriormente como verdad. Esta operación de la API permite aceptar `TransformId` cuyas etiquetas quiere exportar y una ruta de Amazon Simple Storage Service (Amazon S3) en la que exportar las etiquetas. La operación devuelve `TaskRunId`. Puede comprobar el estado de su ejecución de tareas llamando a la API de `GetMLTaskRun`.

Solicitud

- `TransformId`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El identificador único de la transformación de machine learning.
- `OutputS3Path` – Obligatorio: cadena UTF-8.

  La ruta de Amazon S3 en la que exporta las etiquetas.

Respuesta

- `TaskRunId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El identificador único de la ejecución de tareas.

Errores

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

Acción StartImportLabelsTaskRun (Python: start_import_labels_task_run)

Le permite proporcionar etiquetas adicionales (ejemplos de verdad) que se va a utilizar para enseñar a la transformación de machine learning y mejorar su calidad. Esta operación de la API se utiliza normalmente como parte del flujo de trabajo de aprendizaje activo que comienza con la llamada de `StartMLLabelingSetGenerationTaskRun` y que, en última instancia, da lugar a un aumento en la calidad de su transformación de machine learning.

Cuando `StartMLLabelingSetGenerationTaskRun` finaliza, machine learning de AWS Glue dispondrá de una serie de preguntas que los humanos tendrán que responder. (El proceso de respuesta a estas preguntas se suele denominar "etiquetado" en los flujos de trabajo de machine learning). En el caso de la transformación `FindMatches`, estas preguntas tienen la siguiente forma: "¿Cuál es la forma correcta de agrupar estas filas en grupos compuestos enteramente

por registros coincidentes?". Una vez que el proceso de etiquetado finaliza, los usuarios cargan sus respuestas/etiquetas con una llamada a `StartImportLabelsTaskRun`. Una vez que `StartImportLabelsTaskRun` acabe, todas las ejecuciones futuras de la transformación de machine learning utilizarán las etiquetas nuevas y mejoradas, y realizarán una transformación de alta calidad.

De forma predeterminada, `StartMLLabelingSetGenerationTaskRun` aprende continuamente de todas las etiquetas que carga, y las combina, a no ser que establezca `Replace` en true. Si define `Replace` en true, `StartImportLabelsTaskRun` elimina y olvida todas las etiquetas cargadas previamente y aprende solo del conjunto exacto que carga. El reemplazo de etiquetas puede resultar útil si se da cuenta de que ha cargado etiquetas incorrectas anteriormente y cree que tendrán un efecto negativo en la calidad de su transformación.

Puede comprobar el estado de su ejecución de tareas con la operación `GetMLTaskRun`.

Solicitud

- `TransformId`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El identificador único de la transformación de machine learning.
- `InputS3Path` – Obligatorio: cadena UTF-8.

  La ruta de Amazon Simple Storage Service (Amazon S3) desde la que importará las etiquetas.
- `ReplaceAllLabels`: booleano.

  Indica si se van a sobrescribir las etiquetas existentes.

Respuesta

- `TaskRunId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El identificador único de la ejecución de tareas.

Errores

- `EntityNotFoundException`
- `InvalidInputException`

- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`
- `InternalServiceException`

API de Calidad de datos

La API de Calidad de datos describe los tipos de datos de la calidad de los datos e incluye la API para crear, eliminar o actualizar conjuntos de reglas, ejecuciones y evaluadores de la calidad de datos.

Tipos de datos

- DataSource estructura
- DataQualityRulesetListDetails estructura
- DataQualityTargetTable estructura
- DataQualityRulesetEvaluationRunDescription estructura
- DataQualityRulesetEvaluationRunFilter estructura
- DataQualityEvaluationRunAdditionalRunOptions estructura
- DataQualityRuleRecommendationRunDescription estructura
- DataQualityRuleRecommendationRunFilter estructura
- DataQualityResult estructura
- DataQualityAnalyzerResult estructura
- DataQualityObservation estructura
- MetricBasedObservation estructura
- DataQualityMetricValues estructura
- DataQualityRuleResult estructura
- DataQualityResultDescription estructura
- DataQualityResultFilterCriteria estructura
- DataQualityRulesetFilterCriteria estructura

DataSource estructura

Una fuente de datos (una AWS Glue tabla) para la que desea obtener resultados de calidad de datos.

Campos

- `GlueTable`: obligatorio: objeto GlueTable.

  Una AWS Glue tabla.

DataQualityRulesetListDetails estructura

Describe un conjunto de reglas de la calidad de los datos devuelto por `GetDataQualityRuleset`.

Campos

- `Name`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre del conjunto de reglas de calidad de datos.
- `Description`: cadena de descripción de un máximo de 2048 bytes de largo, que coincide con el URI address multi-line string pattern.

  Una descripción del conjunto de reglas de calidad de datos.
- `CreatedOn`: marca temporal.

  La fecha y la hora en que se creó el conjunto de reglas de calidad de datos.
- `LastModifiedOn`: marca temporal.

  La fecha y la hora de la última modificación del conjunto de reglas de calidad de datos.
- `TargetTable`: un objeto DataQualityTargetTable.

  Objeto que representa una AWS Glue tabla.
- `RecommendationRunId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Cuando se crea un conjunto de reglas a partir de una ejecución de recomendación, se genera este ID de ejecución para vincularlos ambos.

- `RuleCount`: número (entero).

  La cantidad de reglas del conjunto de reglas.

DataQualityTargetTable estructura

Objeto que representa una AWS Glue tabla.

Campos

- `TableName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la AWS Glue tabla.
- `DatabaseName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la base de datos en la que existe la AWS Glue tabla.
- `CatalogId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El identificador del catálogo en el que se encuentra la AWS Glue tabla.

DataQualityRulesetEvaluationRunDescription estructura

Describe el resultado de una ejecución de evaluación del conjunto de reglas de calidad de datos.

Campos

- `RunId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El identificador de ejecución único asociado a esta ejecución.
- `Status`: cadena UTF-8 (valores válidos: `STARTING` | `RUNNING` | `STOPPING` | `STOPPED` | `SUCCEEDED` | `FAILED` | `TIMEOUT`).

  El estado de esta ejecución.
- `StartedOn`: marca temporal.

La fecha y la hora en que inició la ejecución.

- `DataSource`: un objeto DataSource.

  La fuente de datos (una AWS Glue tabla) asociada a la ejecución.

DataQualityRulesetEvaluationRunFilter estructura

Los criterios del filtro.

Campos

- `DataSource`: obligatorio: objeto DataSource.

  Filtre en función de una fuente de datos (una AWS Glue tabla) asociada a la ejecución.

- `StartedBefore`: marca temporal.

  Filtre los resultados por las ejecuciones que se iniciaron antes de esta hora.

- `StartedAfter`: marca temporal.

  Filtre los resultados por las ejecuciones que se iniciaron después de esta hora.

DataQualityEvaluationRunAdditionalRunOptions estructura

Opciones de ejecución adicionales que puede especificar para una ejecución de evaluación.

Campos

- `CloudWatchMetricsEnabled`: booleano.

  Habilitar o no CloudWatch las métricas.

- `ResultsS3Prefix`: cadena UTF-8.

  Prefijo para que Amazon S3 almacene los resultados.

- `CompositeRuleEvaluationMethod`: cadena UTF-8 (valores válidos: `COLUMN` | `ROW`).

  Establezca el método de evaluación para las reglas compuestas del conjunto de reglas en ROW/COLUMN

DataQualityRuleRecommendationRunDescription estructura

Describe el resultado de una ejecución de recomendación de reglas de la calidad de los datos.

Campos

- `RunId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El identificador de ejecución único asociado a esta ejecución.
- `Status`: cadena UTF-8 (valores válidos: `STARTING` | `RUNNING` | `STOPPING` | `STOPPED` | `SUCCEEDED` | `FAILED` | `TIMEOUT`).

  El estado de esta ejecución.
- `StartedOn`: marca temporal.

  La fecha y la hora en las que se inició esta ejecución.
- `DataSource`: un objeto DataSource.

  La fuente de datos (AWS Glue tabla) asociada a la ejecución de recomendaciones.

DataQualityRuleRecommendationRunFilter estructura

Se ejecuta un filtro para enumerar las ejecuciones de recomendación de la calidad de los datos.

Campos

- `DataSource`: obligatorio: objeto DataSource.

  Filtrar en función de una fuente de datos específica (AWS Glue tabla).
- `StartedBefore`: marca temporal.

  Filtre en función de la hora de los resultados que se iniciaron antes de la hora indicada.
- `StartedAfter`: marca temporal.

  Filtre en función de la hora de los resultados que se iniciaron después de la hora indicada.

DataQualityResult estructura

Describe un resultado de la calidad de los datos.

Campos

- `ResultId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Un identificador de resultado único para el resultado de la calidad de los datos.
- `Score`: número (doble), 1,0 como máximo.

  Una puntuación agregada de la calidad de los datos. Representa la relación entre las reglas aprobadas y la cantidad total de reglas.
- `DataSource`: un objeto DataSource.

  La tabla asociada al resultado de calidad de datos, si existe.
- `RulesetName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre del conjunto de reglas asociado al resultado de calidad de datos.
- `EvaluationContext`: cadena UTF-8.

  En el contexto de un trabajo en AWS Glue Studio, a cada nodo del lienzo se le suele asignar algún tipo de nombre y los nodos de calidad de los datos tendrán nombres. En el caso de varios nodos, `evaluationContext` puede diferenciar los nodos.
- `StartedOn`: marca temporal.

  La fecha y la hora en las que se inició esta ejecución de la calidad de los datos.
- `CompletedOn`: marca temporal.

  La fecha y la hora en las que finalizó esta ejecución de la calidad de los datos.
- `JobName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre del trabajo asociado al resultado de calidad de datos, si existe.
- `JobRunId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

El ID de la ejecución del trabajo asociado al resultado de calidad de datos, si existe.

- `RulesetEvaluationRunId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID único de la ejecución para la evaluación del conjunto de reglas de este resultado de calidad de datos.

- `RuleResults`: matriz de objetos DataQualityRuleResult, con 2000 estructuras como máximo.

  Una lista de objetos `DataQualityRuleResult` que representan los resultados de cada regla.

- `AnalyzerResults`: matriz de objetos DataQualityAnalyzerResult, con 2000 estructuras como máximo.

  Una lista de objetos `DataQualityAnalyzerResult` que representan los resultados de cada analizador.

- `Observations`: matriz de objetos DataQualityObservation, con 50 estructuras como máximo.

  Una lista de objetos `DataQualityObservation` que representan la observabilidad generada después de evaluar las reglas y los analizadores.

DataQualityAnalyzerResult estructura

Describe el resultado de la evaluación del analizador de la calidad de datos.

Campos

- `Name`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre del analizador de la calidad de datos.

- `Description`: cadena UTF-8 con un máximo de 2048 bytes de largo, que coincide con el URI address multi-line string pattern.

  Descripción del analizador de la calidad de datos.

- `EvaluationMessage`: cadena UTF-8 con un máximo de 2048 bytes de largo, que coincide con el URI address multi-line string pattern.

  Un mensaje de evaluación.

- `EvaluatedMetrics`: matriz de mapas de pares clave-valor.

  Cada clave es una cadena UTF-8 con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Cada valor es un número (doble).

  Un mapa de las métricas asociadas a la evaluación del analizador.

DataQualityObservation estructura

Describe la observación generada luego de la evaluación de las reglas y los analizadores.

Campos

- `Description`: cadena UTF-8 con un máximo de 2048 bytes de largo, que coincide con el URI address multi-line string pattern.

  Una descripción de la observación de la calidad de datos.
- `MetricBasedObservation`: un objeto MetricBasedObservation.

  Un objeto de tipo `MetricBasedObservation` que representa la observación basada en las métricas evaluadas de la calidad de datos.

MetricBasedObservation estructura

Describe la observación generada según las métricas basada en las métricas evaluadas de la calidad de datos.

Campos

- `MetricName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la métrica de calidad de datos utilizada para la generación de la observación.
- `MetricValues`: un objeto DataQualityMetricValues.

  Un objeto de tipo `DataQualityMetricValues` que representa el análisis del valor de la métrica de la calidad de datos.

- `NewRules`: matriz de cadenas UTF-8.

  Una lista de reglas de calidad de datos nuevas generadas como parte de la observación basada en el valor de la métrica de la calidad de datos.

DataQualityMetricValues estructura

Describe el valor de la métrica de la calidad de datos según el análisis de los datos históricos.

Campos

- `ActualValue`: número (doble).

  El valor real de la métrica de la calidad de datos.
- `ExpectedValue`: número (doble).

  El valor esperado de la métrica de la calidad de datos según el análisis de los datos históricos.
- `LowerLimit`: número (doble).

  El valor más bajo del valor de la métrica de la calidad de datos según el análisis de los datos históricos.
- `UpperLimit`: número (doble).

  El valor más alto del valor de la métrica de la calidad de datos según el análisis de los datos históricos.

DataQualityRuleResult estructura

Describe el resultado de la evaluación de la regla de la calidad de datos.

Campos

- `Name`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de la regla de la calidad de datos.
- `Description`: cadena UTF-8 con un máximo de 2048 bytes de largo, que coincide con el URI address multi-line string pattern.

Descripción de la regla de la calidad de datos.

- `EvaluationMessage`: cadena UTF-8 con un máximo de 2048 bytes de largo, que coincide con el URI address multi-line string pattern.

  Un mensaje de evaluación.

- `Result`: cadena UTF-8 (valores válidos: `PASS` | `FAIL` | `ERROR`).

  Estado de aprobación o rechazo de la regla.

- `EvaluatedMetrics`: matriz de mapas de pares clave-valor.

  Cada clave es una cadena UTF-8 con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Cada valor es un número (doble).

  Un mapa de métricas asociadas a la evaluación de la regla.

DataQualityResultDescription estructura

Describe un resultado de la calidad de los datos.

Campos

- `ResultId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El identificador único de este resultado de la calidad de datos.

- `DataSource`: un objeto DataSource.

  El nombre de la tabla asociada al resultado de la calidad de datos.

- `JobName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre del trabajo asociado al resultado de la calidad de datos.

- `JobRunId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El identificador de la ejecución del trabajo asociado al resultado de la calidad de datos.

- `StartedOn`: marca temporal.

  La hora a la que se inició la ejecución de este resultado de la calidad de datos.

DataQualityResultFilterCriteria estructura

Criterios utilizados para devolver los resultados de la calidad de los datos.

Campos

- `DataSource`: un objeto DataSource.

  Filtre los resultados por el origen de datos especificado. Por ejemplo, recuperar todos los resultados de una AWS Glue tabla.
- `JobName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Filtre los resultados por el nombre de trabajo indicado.
- `JobRunId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Filtre los resultados por el identificador de la ejecución de trabajo indicado.
- `StartedAfter`: marca temporal.

  Filtre los resultados por las ejecuciones que se iniciaron después de esta hora.
- `StartedBefore`: marca temporal.

  Filtre los resultados por las ejecuciones que se iniciaron antes de esta hora.

DataQualityRulesetFilterCriteria estructura

Criterios utilizados para filtrar los conjuntos de reglas de calidad de datos.

Campos

- `Name`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre de los criterios del filtro del conjunto de reglas.

- `Description`: cadena de descripción de un máximo de 2048 bytes de largo, que coincide con el URI address multi-line string pattern.

  La descripción de los criterios del filtro del conjunto de reglas.

- `CreatedBefore`: marca temporal.

  Filtre los conjuntos de reglas creados antes de esta fecha.

- `CreatedAfter`: marca temporal.

  Filtre los conjuntos de reglas creados después de esta fecha.

- `LastModifiedBefore`: marca temporal.

  Filtre los últimos conjuntos de reglas modificados antes de esta fecha.

- `LastModifiedAfter`: marca temporal.

  Filtre los últimos conjuntos de reglas modificados después de esta fecha.

- `TargetTable`: un objeto DataQualityTargetTable.

  El nombre de la base de datos y el nombre de la tabla de destino.

Operaciones

- StartDataQualityRulesetEvaluationRun acción (Python: start_data_quality_ruleset_evaluation_run)
- CancelDataQualityRulesetEvaluationRun acción (Python: cancel_data_quality_ruleset_evaluation_run)
- GetDataQualityRulesetEvaluationRun acción (Python: get_data_quality_ruleset_evaluation_run)
- ListDataQualityRulesetEvaluationRuns acción (Python: list_data_quality_ruleset_evaluation_runs)
- StartDataQualityRuleRecommendationRun acción (Python: start_data_quality_rule_recommendation_run)
- CancelDataQualityRuleRecommendationRun acción (Python: cancel_data_quality_rule_recommendation_run)
- GetDataQualityRuleRecommendationRun acción (Python: get_data_quality_rule_recommendation_run)
- ListDataQualityRuleRecommendationRuns acción (Python: list_data_quality_rule_recommendation_runs)

- GetDataQualityResult acción (Python: get_data_quality_result)
- BatchGetDataQualityResult acción (Python: batch_get_data_quality_result)
- ListDataQualityResults acción (Python: list_data_quality_results)
- CreateDataQualityRuleset acción (Python: create_data_quality_ruleset)
- DeleteDataQualityRuleset acción (Python: delete_data_quality_ruleset)
- GetDataQualityRuleset acción (Python: get_data_quality_ruleset)
- ListDataQualityRulesets acción (Python: list_data_quality_rulesets)
- UpdateDataQualityRuleset acción (Python: update_data_quality_ruleset)

StartDataQualityRulesetEvaluationRun acción (Python: start_data_quality_ruleset_evaluation_run)

Una vez que tenga una definición de conjunto de reglas (recomendada o propia), llame a esta operación para evaluar el conjunto de reglas con respecto a una fuente de datos (tabla).AWS Glue La evaluación calcula los resultados que puede recuperar con la API de `GetDataQualityResult`.

Solicitud

- `DataSource`: obligatorio: objeto DataSource.

  La fuente de datos (AWS Glue tabla) asociada a esta ejecución.
- `Role` – Obligatorio: cadena UTF-8.

  IAM Función proporcionada para cifrar los resultados de la ejecución.
- `NumberOfWorkers`: número (entero).

  La cantidad de procesos de trabajo de `G.1X` utilizados para la ejecución. El valor predeterminado es 5.
- `Timeout`: número (entero), como mínimo 1.

  El tiempo de espera durante una ejecución en minutos. Es el tiempo máximo que una ejecución puede consumir recursos antes de que se termine y cambie al estado `TIMEOUT`. El valor predeterminado es 2880 minutos (48 horas).
- `ClientToken`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

Se usa para la idempotencia y se recomienda establecerlo en un identificador aleatorio (como un UUID) para evitar crear o iniciar varias instancias del mismo recurso.

- `AdditionalRunOptions`: un objeto DataQualityEvaluationRunAdditionalRunOptions.

  Opciones de ejecución adicionales que puede especificar para una ejecución de evaluación.

- `RulesetNames`: obligatorio: matriz de cadenas UTF-8, con una cadena como mínimo y 10 cadenas como máximo.

  Lista de nombres de conjuntos de reglas.

- `AdditionalDataSources`: matriz de mapas de pares clave-valor.

  Cada clave es una cadena UTF-8 con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Cada valor es un objeto A DataSource.

  Un mapa de cadenas de referencia a orígenes de datos adicionales que puede especificar para una ejecución de evaluación.

Respuesta

- `RunId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El identificador de ejecución único asociado a esta ejecución.

Errores

- `InvalidInputException`
- `EntityNotFoundException`
- `OperationTimeoutException`
- `InternalServiceException`
- `ConflictException`

CancelDataQualityRulesetEvaluationRun acción (Python: cancel_data_quality_ruleset_evaluation_run)

Cancela una ejecución en la que se evalúa un conjunto de reglas con respecto a un origen de datos.

Solicitud

- `RunId`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El identificador de ejecución único asociado a esta ejecución.

Respuesta

- Sin parámetros de respuesta.

Errores

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

GetDataQualityRulesetEvaluationRun acción (Python: get_data_quality_ruleset_evaluation_run)

Recupera una ejecución específica en la que se evalúa un conjunto de reglas con respecto a un origen de datos.

Solicitud

- `RunId`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El identificador de ejecución único asociado a esta ejecución.

Respuesta

- `RunId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El identificador de ejecución único asociado a esta ejecución.

- `DataSource`: un objeto DataSource.

  La fuente de datos (una tabla) asociada a esta ejecución de evaluación. AWS Glue

- `Role`: cadena UTF-8.

  IAM Función proporcionada para cifrar los resultados de la ejecución.

- `NumberOfWorkers`: número (entero).

  La cantidad de procesos de trabajo de `G.1X` utilizados para la ejecución. El valor predeterminado es 5.

- `Timeout`: número (entero), como mínimo 1.

  El tiempo de espera durante una ejecución en minutos. Es el tiempo máximo que una ejecución puede consumir recursos antes de que se termine y cambie al estado `TIMEOUT`. El valor predeterminado es 2880 minutos (48 horas).

- `AdditionalRunOptions`: un objeto DataQualityEvaluationRunAdditionalRunOptions.

  Opciones de ejecución adicionales que puede especificar para una ejecución de evaluación.

- `Status`: cadena UTF-8 (valores válidos: `STARTING` | `RUNNING` | `STOPPING` | `STOPPED` | `SUCCEEDED` | `FAILED` | `TIMEOUT`).

  El estado de esta ejecución.

- `ErrorString`: cadena UTF-8.

  Las cadenas de error asociadas a la ejecución.

- `StartedOn`: marca temporal.

  La fecha y la hora en las que se inició esta ejecución.

- `LastModifiedOn`: marca temporal.

  Una marca temporal. El último momento dado en el que se modificó esta ejecución de recomendación de la regla de calidad de datos.

- `CompletedOn`: marca temporal.

  La fecha y la hora en las que se completó esta ejecución.
- `ExecutionTime`: número (entero).

  El tiempo (en segundos) que la ejecución de flujo de trabajo consumió recursos.
- `RulesetNames`: matriz de cadenas UTF-8, con una cadena como mínimo y 10 cadenas como máximo.

  Lista de los nombres de los conjuntos de reglas para la ejecución. Actualmente, este parámetro solo puede tener un nombre para el conjunto de reglas.
- `ResultIds`: matriz de cadenas UTF-8, con una cadena como mínimo y 10 cadenas como máximo.

  Una lista de ID de los resultados de calidad de datos de la ejecución.
- `AdditionalDataSources`: matriz de mapas de pares clave-valor.

  Cada clave es una cadena UTF-8 con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Cada valor es un objeto A DataSource.

  Un mapa de cadenas de referencia a orígenes de datos adicionales que puede especificar para una ejecución de evaluación.

Errores

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

ListDataQualityRulesetEvaluationRuns acción (Python: list_data_quality_ruleset_evaluation_runs)

Muestra todas las ejecuciones que cumplen los criterios del filtro, donde un conjunto de reglas se evalúa en relación con un origen de datos.

Solicitud

- `Filter`: un objeto DataQualityRulesetEvaluationRunFilter.

  Los criterios del filtro.
- `NextToken`: cadena UTF-8.

  Un token paginado para compensar los resultados.
- `MaxResults`: número (entero) que no es inferior a 1 ni es superior a 1000.

  El número máximo de resultados que devolver.

Respuesta

- `Runs`: matriz de objetos DataQualityRulesetEvaluationRunDescription.

  Una lista de objetos `DataQualityRulesetEvaluationRunDescription` que representa las ejecuciones de un conjunto de reglas de calidad de datos.
- `NextToken`: cadena UTF-8.

  Un token de paginación, si hay disponibles más resultados.

Errores

- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

StartDataQualityRuleRecommendationRun acción (Python: start_data_quality_rule_recommendation_run)

Inicia una serie de recomendaciones que se utiliza para generar reglas cuando no se sabe qué reglas escribir. AWS Glue Data Quality analiza los datos y presenta recomendaciones para un posible conjunto de reglas. A continuación, puede clasificar el conjunto de reglas y modificar el conjunto de reglas generado a su gusto.

Las ejecuciones de recomendaciones se eliminan automáticamente después de 90 días.

Solicitud

- `DataSource`: obligatorio: objeto DataSource.

  La fuente de datos (AWS Glue tabla) asociada a esta ejecución.
- `Role` – Obligatorio: cadena UTF-8.

  IAM Función proporcionada para cifrar los resultados de la ejecución.
- `NumberOfWorkers`: número (entero).

  La cantidad de procesos de trabajo de `G.1X` utilizados para la ejecución. El valor predeterminado es 5.
- `Timeout`: número (entero), como mínimo 1.

  El tiempo de espera durante una ejecución en minutos. Es el tiempo máximo que una ejecución puede consumir recursos antes de que se termine y cambie al estado `TIMEOUT`. El valor predeterminado es 2880 minutos (48 horas).
- `CreatedRulesetName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Un nombre para el conjunto de reglas.
- `ClientToken`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Se usa para la idempotencia y se recomienda establecerlo en un identificador aleatorio (como un UUID) para evitar crear o iniciar varias instancias del mismo recurso.

Respuesta

- `RunId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El identificador de ejecución único asociado a esta ejecución.

Errores

- `InvalidInputException`
- `OperationTimeoutException`

- `InternalServiceException`
- `ConflictException`

CancelDataQualityRuleRecommendationRun acción (Python: cancel_data_quality_rule_recommendation_run)

Cancela la ejecución de recomendación especificada que se utilizó para generar las reglas.

Solicitud

- `RunId`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El identificador de ejecución único asociado a esta ejecución.

Respuesta

- Sin parámetros de respuesta.

Errores

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

GetDataQualityRuleRecommendationRun acción (Python: get_data_quality_rule_recommendation_run)

Obtiene la ejecución de recomendación especificada que se utilizó para generar las reglas.

Solicitud

- `RunId`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El identificador de ejecución único asociado a esta ejecución.

Respuesta

- `RunId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El identificador de ejecución único asociado a esta ejecución.

- `DataSource`: un objeto DataSource.

  La fuente de datos (una tabla) asociada a esta ejecución. AWS Glue

- `Role`: cadena UTF-8.

  IAM Función proporcionada para cifrar los resultados de la ejecución.

- `NumberOfWorkers`: número (entero).

  La cantidad de procesos de trabajo de `G.1X` utilizados para la ejecución. El valor predeterminado es 5.

- `Timeout`: número (entero), como mínimo 1.

  El tiempo de espera durante una ejecución en minutos. Es el tiempo máximo que una ejecución puede consumir recursos antes de que se termine y cambie al estado `TIMEOUT`. El valor predeterminado es 2880 minutos (48 horas).

- `Status`: cadena UTF-8 (valores válidos: `STARTING` | `RUNNING` | `STOPPING` | `STOPPED` | `SUCCEEDED` | `FAILED` | `TIMEOUT`).

  El estado de esta ejecución.

- `ErrorString`: cadena UTF-8.

  Las cadenas de error asociadas a la ejecución.

- `StartedOn`: marca temporal.

  La fecha y la hora en las que se inició esta ejecución.

- `LastModifiedOn`: marca temporal.

  Una marca temporal. El último momento dado en el que se modificó esta ejecución de recomendación de la regla de calidad de datos.

- `CompletedOn`: marca temporal.

  La fecha y la hora en las que se completó esta ejecución.

- `ExecutionTime`: número (entero).

  El tiempo (en segundos) que la ejecución de flujo de trabajo consumió recursos.
- `RecommendedRuleset`: cadena UTF-8, con 1 byte de largo como mínimo o 65 536 como máximo.

  Cuando se completa una ejecución de recomendación de reglas de inicio, se crea un conjunto de reglas recomendado (un conjunto de reglas). Este miembro tiene esas reglas en formato de lenguaje de definición de calidad de datos (DQDL).
- `CreatedRulesetName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre del conjunto de reglas que se creó durante la ejecución.

Errores

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

ListDataQualityRuleRecommendationRuns acción (Python: list_data_quality_rule_recommendation_runs)

Muestra las ejecuciones de recomendación que cumplen los criterios del filtro.

Solicitud

- `Filter`: un objeto DataQualityRuleRecommendationRunFilter.

  Los criterios del filtro.
- `NextToken`: cadena UTF-8.

  Un token paginado para compensar los resultados.
- `MaxResults`: número (entero) que no es inferior a 1 ni es superior a 1000.

  El número máximo de resultados que devolver.

Respuesta

- `Runs`: matriz de objetos DataQualityRuleRecommendationRunDescription.

  Una lista de objetos `DataQualityRuleRecommendationRunDescription`.

- `NextToken`: cadena UTF-8.

  Un token de paginación, si hay disponibles más resultados.

Errores

- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

GetDataQualityResult acción (Python: get_data_quality_result)

Recupera el resultado de una evaluación de la regla de calidad de datos.

Solicitud

- `ResultId`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Un identificador de resultado único para el resultado de la calidad de los datos.

Respuesta

- `ResultId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Un identificador de resultado único para el resultado de la calidad de los datos.

- `Score`: número (doble), 1,0 como máximo.

  Una puntuación agregada de la calidad de los datos. Representa la relación entre las reglas aprobadas y la cantidad total de reglas.

- `DataSource`: un objeto DataSource.

La tabla asociada al resultado de calidad de datos, si existe.

- `RulesetName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre del conjunto de reglas asociado al resultado de calidad de datos.

- `EvaluationContext`: cadena UTF-8.

  En el contexto de un trabajo en AWS Glue Studio, a cada nodo del lienzo se le suele asignar algún tipo de nombre y los nodos de calidad de los datos tendrán nombres. En el caso de varios nodos, `evaluationContext` puede diferenciar los nodos.

- `StartedOn`: marca temporal.

  La fecha y la hora en las que se inició la ejecución de este resultado de calidad de datos.

- `CompletedOn`: marca temporal.

  La fecha y la hora en las que se completó la ejecución de este resultado de calidad de datos.

- `JobName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre del trabajo asociado al resultado de calidad de datos, si existe.

- `JobRunId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID de la ejecución del trabajo asociado al resultado de calidad de datos, si existe.

- `RulesetEvaluationRunId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID único de ejecución asociado a la evaluación del conjunto de reglas.

- `RuleResults`: matriz de objetos DataQualityRuleResult, con 2000 estructuras como máximo.

  Una lista de objetos `DataQualityRuleResult` que representan los resultados de cada regla.

- `AnalyzerResults`: matriz de objetos DataQualityAnalyzerResult, con 2000 estructuras como máximo.

  Una lista de objetos `DataQualityAnalyzerResult` que representan los resultados de cada analizador.

- `Observations`: matriz de objetos DataQualityObservation, con 50 estructuras como máximo.

Una lista de objetos `DataQualityObservation` que representan la observabilidad generada después de evaluar las reglas y los analizadores.

Errores

- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`
- `EntityNotFoundException`

BatchGetDataQualityResult acción (Python: batch_get_data_quality_result)

Recupera una lista de los resultados de calidad de datos para los ID de los resultados especificados.

Solicitud

- `ResultIds`: obligatorio: matriz de cadenas UTF-8, con una cadena como mínimo y 100 cadenas como máximo.

  Una lista de identificadores únicos de los resultados de los resultados de calidad de datos.

Respuesta

- `Results` (obligatorio): una matriz de objetos DataQualityResult.

  Una lista de objetos `DataQualityResult` que representa los resultados de calidad de datos.
- `ResultsNotFound`: matriz de cadenas UTF-8, con una cadena como mínimo y 100 cadenas como máximo.

  Una lista de ID de los resultados para los que no se encontraron resultados.

Errores

- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

ListDataQualityResults acción (Python: list_data_quality_results)

Devuelve todos los resultados de la ejecución de calidad de datos de su cuenta.

Solicitud

- `Filter`: un objeto DataQualityResultFilterCriteria.

  Los criterios del filtro.
- `NextToken`: cadena UTF-8.

  Un token paginado para compensar los resultados.
- `MaxResults`: número (entero) que no es inferior a 1 ni es superior a 1000.

  El número máximo de resultados que devolver.

Respuesta

- `Results` (obligatorio): una matriz de objetos DataQualityResultDescription.

  Una lista de objetos `DataQualityResultDescription`.
- `NextToken`: cadena UTF-8.

  Un token de paginación, si hay disponibles más resultados.

Errores

- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

CreateDataQualityRuleset acción (Python: create_data_quality_ruleset)

Crea un conjunto de reglas de calidad de datos con reglas de DQDL aplicadas a una tabla específica. AWS Glue

El conjunto de reglas se crea mediante lenguaje de definición de calidad de datos (DQDL). Para obtener más información, consulte la guía para desarrolladores. AWS Glue

Solicitud

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre único para el conjunto de reglas de la calidad de los datos.

- `Description`: cadena de descripción de un máximo de 2048 bytes de largo, que coincide con el URI address multi-line string pattern.

  Una descripción del conjunto de reglas de calidad de datos.

- `Ruleset`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 65 536 como máximo.

  Conjunto de reglas de lenguaje de definición de calidad de datos (DQDL). Para obtener más información, consulta la guía para AWS Glue desarrolladores.

- `Tags`: matriz de mapas de pares de clave-valor, con 50 pares como máximo.

  Cada clave es una cadena UTF-8 con una longitud de entre 1 y 128 bytes.

  Cada valor es una cadena UTF-8 que no tiene más de 256 bytes de largo.

  Una lista de etiquetas aplicadas al conjunto de reglas de calidad de datos.

- `TargetTable`: un objeto DataQualityTargetTable.

  Una tabla de destino asociada al conjunto de reglas de calidad de datos.

- `RecommendationRunId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Un identificador único de ejecución para la ejecución de recomendación.

- `ClientToken`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Se usa para la idempotencia y se recomienda establecerlo en un identificador aleatorio (como un UUID) para evitar crear o iniciar varias instancias del mismo recurso.

Respuesta

- `Name`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

Nombre único para el conjunto de reglas de la calidad de los datos.

Errores

- `InvalidInputException`
- `AlreadyExistsException`
- `OperationTimeoutException`
- `InternalServiceException`
- `ResourceNumberLimitExceededException`

DeleteDataQualityRuleset acción (Python: delete_data_quality_ruleset)

Elimina un conjunto de reglas de la calidad de los datos.

Solicitud

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Un nombre para el conjunto de reglas de calidad de datos.

Respuesta

- Sin parámetros de respuesta.

Errores

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

GetDataQualityRuleset acción (Python: get_data_quality_ruleset)

Devuelve un conjunto de reglas existente por identificador o nombre.

Solicitud

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre del conjunto de reglas.

Respuesta

- `Name`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre del conjunto de reglas.
- `Description`: cadena de descripción de un máximo de 2048 bytes de largo, que coincide con el URI address multi-line string pattern.

  Una descripción del conjunto de reglas.
- `Ruleset`: cadena UTF-8, con 1 byte de largo como mínimo o 65 536 como máximo.

  Conjunto de reglas de lenguaje de definición de calidad de datos (DQDL). Para obtener más información, consulta la guía para desarrolladores. AWS Glue
- `TargetTable`: un objeto DataQualityTargetTable.

  El nombre de la base de datos y el nombre de la tabla de destino.
- `CreatedOn`: marca temporal.

  Una marca temporal. La fecha y la hora en las que se creó este conjunto de reglas de calidad de datos.
- `LastModifiedOn`: marca temporal.

  Una marca temporal. El último momento dado en el que se modificó este conjunto de reglas de calidad de datos.
- `RecommendationRunId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Cuando se crea un conjunto de reglas a partir de una ejecución de recomendación, se genera este ID de ejecución para vincularlos ambos.

Errores

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

ListDataQualityRulesets acción (Python: list_data_quality_rulesets)

Devuelve una lista paginada de conjuntos de reglas para la lista de tablas especificada. AWS Glue

Solicitud

- `NextToken`: cadena UTF-8.

  Un token paginado para compensar los resultados.
- `MaxResults`: número (entero) que no es inferior a 1 ni es superior a 1000.

  El número máximo de resultados que devolver.
- `Filter`: un objeto DataQualityRulesetFilterCriteria.

  Los criterios del filtro.
- `Tags`: matriz de mapas de pares de clave-valor, con 50 pares como máximo.

  Cada clave es una cadena UTF-8 con una longitud de entre 1 y 128 bytes.

  Cada valor es una cadena UTF-8 que no tiene más de 256 bytes de largo.

  Una lista de etiquetas de par clave-valor.

Respuesta

- `Rulesets`: matriz de objetos DataQualityRulesetListDetails.

  Una lista paginada de conjuntos de reglas para la lista de tablas especificada. AWS Glue
- `NextToken`: cadena UTF-8.

  Un token de paginación, si hay disponibles más resultados.

Errores

- `EntityNotFoundException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

UpdateDataQualityRuleset acción (Python: update_data_quality_ruleset)

Actualiza el conjunto de reglas de calidad de datos especificado.

Solicitud

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre del conjunto de reglas de calidad de datos.
- `Description`: cadena de descripción de un máximo de 2048 bytes de largo, que coincide con el URI address multi-line string pattern.

  Una descripción del conjunto de reglas.
- `Ruleset`: cadena UTF-8, con 1 byte de largo como mínimo o 65 536 como máximo.

  Conjunto de reglas de lenguaje de definición de calidad de datos (DQDL). Para obtener más información, consulta la guía para desarrolladores. AWS Glue

Respuesta

- `Name`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre del conjunto de reglas de calidad de datos.
- `Description`: cadena de descripción de un máximo de 2048 bytes de largo, que coincide con el URI address multi-line string pattern.

  Una descripción del conjunto de reglas.
- `Ruleset`: cadena UTF-8, con 1 byte de largo como mínimo o 65 536 como máximo.

Conjunto de reglas de lenguaje de definición de calidad de datos (DQDL). Para obtener más información, consulta la guía para AWS Glue desarrolladores.

Errores

- `EntityNotFoundException`
- `AlreadyExistsException`
- `IdempotentParameterMismatchException`
- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`
- `ResourceNumberLimitExceededException`

API de detección de información confidencial

La API de detección de información confidencial describe las API utilizadas para detectar información confidencial en las columnas y filas de los datos estructurados.

Tipos de datos

- Estructura CustomEntityType

Estructura CustomEntityType

Objeto que representa un patrón personalizado para detectar información confidencial en las columnas y filas de los datos estructurados.

Campos

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre del patrón personalizado que permite recuperarlo o eliminarlo más adelante. Este nombre debe ser único por cuenta de AWS.

- `RegexString`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Cadena de expresión regular que se utiliza para detectar información confidencial en un patrón personalizado.

- `ContextWords`: matriz de cadenas UTF-8, con una cadena como mínimo o más de 20 cadenas.

  Lista de palabras contextuales. Si no se encuentra ninguna de estas palabras contextuales en las proximidades de la expresión regular, los datos no se detectarán como información confidencial.

  Si no se pasan palabras contextuales, solo se comprueba una expresión regular.

Operaciones

- Acción CreateCustomEntityType (Python: create_custom_entity_type)
- Acción DeleteCustomEntityType (Python: delete_custom_entity_type)
- Acción GetCustomEntityType (Python: get_custom_entity_type)
- Acción BatchGetCustomEntityTypes (Python: batch_get_custom_entity_types)
- Acción ListCustomEntityTypes (Python: list_custom_entity_types)

Acción CreateCustomEntityType (Python: create_custom_entity_type)

Crea un patrón personalizado que se utiliza para detectar información confidencial en las columnas y filas de los datos estructurados.

Cada patrón personalizado que cree especifica una expresión regular y una lista opcional de palabras contextuales. Si no se pasan palabras contextuales, solo se comprueba una expresión regular.

Solicitud

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre del patrón personalizado que permite recuperarlo o eliminarlo más adelante. Este nombre debe ser único por cuenta de AWS.

- `RegexString`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Cadena de expresión regular que se utiliza para detectar información confidencial en un patrón personalizado.

- `ContextWords`: matriz de cadenas UTF-8, con una cadena como mínimo o más de 20 cadenas.

  Lista de palabras contextuales. Si no se encuentra ninguna de estas palabras contextuales en las proximidades de la expresión regular, los datos no se detectarán como información confidencial.

  Si no se pasan palabras contextuales, solo se comprueba una expresión regular.

- `Tags`: matriz de mapas de pares de clave-valor, con 50 pares como máximo.

  Cada clave es una cadena UTF-8 con una longitud de entre 1 y 128 bytes.

  Cada valor es una cadena UTF-8 que no tiene más de 256 bytes de largo.

  Una lista de etiquetas aplicadas al tipo de entidad personalizada.

Respuesta

- `Name`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre del patrón personalizado que ha creado.

Errores

- `AccessDeniedException`
- `AlreadyExistsException`
- `IdempotentParameterMismatchException`
- `InternalServiceException`
- `InvalidInputException`
- `OperationTimeoutException`
- `ResourceNumberLimitExceededException`

Acción DeleteCustomEntityType (Python: delete_custom_entity_type)

Elimina un patrón personalizado especificando su nombre.

Solicitud

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre del patrón personalizado que desea eliminar.

Respuesta

- `Name`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre del patrón personalizado que ha eliminado.

Errores

- `EntityNotFoundException`
- `AccessDeniedException`
- `InternalServiceException`
- `InvalidInputException`
- `OperationTimeoutException`

Acción GetCustomEntityType (Python: get_custom_entity_type)

Recupera los detalles de un patrón personalizado especificando su nombre.

Solicitud

- `Name`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre del patrón personalizado que desea recuperar.

Respuesta

- `Name`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El nombre del patrón personalizado que ha recuperado.

- `RegexString`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Cadena de expresión regular que se utiliza para detectar información confidencial en un patrón personalizado.

- `ContextWords`: matriz de cadenas UTF-8, con una cadena como mínimo o más de 20 cadenas.

  Lista de palabras contextuales si se especifica al crear el patrón personalizado. Si no se encuentra ninguna de estas palabras contextuales en las proximidades de la expresión regular, los datos no se detectarán como información confidencial.

Errores

- `EntityNotFoundException`
- `AccessDeniedException`
- `InternalServiceException`
- `InvalidInputException`
- `OperationTimeoutException`

Acción BatchGetCustomEntityTypes (Python: batch_get_custom_entity_types)

Recupera los detalles de los patrones personalizados especificados por una lista de nombres.

Solicitud

- `Names`: Obligatorio: matriz de cadenas UTF-8, con una cadena como mínimo y 50 cadenas como máximo.

  Una lista de nombres de patrones personalizados que desea recuperar.

Respuesta

- `CustomEntityTypes`: matriz de objetos CustomEntityType.

  Una lista de objetos `CustomEntityType` que representan los patrones personalizados que se han creado.
- `CustomEntityTypesNotFound`: matriz de cadenas UTF-8, con una cadena como mínimo o más de 50 cadenas.

  Una lista de los nombres de patrones personalizados que no se han encontrado.

Errores

- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`

Acción ListCustomEntityTypes (Python: list_custom_entity_types)

Enumera todos los patrones personalizados que se han creado.

Solicitud

- `NextToken`: cadena UTF-8.

  Un token paginado para compensar los resultados.
- `MaxResults`: número (entero) que no es inferior a 1 ni es superior a 1000.

  El número máximo de resultados que devolver.
- `Tags`: matriz de mapas de pares de clave-valor, con 50 pares como máximo.

  Cada clave es una cadena UTF-8 con una longitud de entre 1 y 128 bytes.

  Cada valor es una cadena UTF-8 que no tiene más de 256 bytes de largo.

  Una lista de etiquetas de par clave-valor.

Respuesta

- `CustomEntityTypes`: matriz de objetos CustomEntityType.

  Una lista de objetos `CustomEntityType` que representan patrones personalizados.
- `NextToken`: cadena UTF-8.

  Un token de paginación, si hay disponibles más resultados.

Errores

- `InvalidInputException`
- `OperationTimeoutException`
- `InternalServiceException`

API de etiquetado en AWS Glue

Tipos de datos

- Estructura de etiquetas

Estructura de etiquetas

El objeto `Tag` representa una etiqueta que se puede asignar a un recurso de AWS. Cada etiqueta está formada por una clave y un valor opcional, ambos definidos por el usuario.

Para obtener más información acerca de las etiquetas y controlar el acceso a los recursos en AWS Glue, consulte Etiquetas de AWS en AWS Glue y Especificación de ARN de recursos de AWS Glue en la Guía para desarrolladores.

Campos

- `key`: cadena UTF-8, con 1 byte de largo como mínimo y 128 bytes de largo como máximo.

  La clave de la etiqueta. La clave es necesaria para crear una etiqueta en un objeto. La clave distingue entre mayúsculas y minúsculas y no debe contener el prefijo aws.
- `value`: cadena UTF-8 de 256 bytes de largo como máximo.

El valor de la etiqueta. El valor es opcional para crear una etiqueta en un objeto. El valor distingue entre mayúsculas y minúsculas y no debe contener el prefijo aws.

Operaciones

- Acción TagResource (Python: tag_resource)
- Acción UntagResource (Python: untag_resource)
- Acción GetTags (Python: get_tags)

Acción TagResource (Python: tag_resource)

Agrega etiquetas a un recurso. Una etiqueta es una marca que se puede asignar a un recurso de AWS. En AWS Glue, solamente se pueden etiquetar ciertos recursos. Para obtener más información acerca de qué recursos pueden etiquetarse, consulte Etiquetas de AWS en AWS Glue.

Además de los permisos de etiquetado para llamar a las API relacionadas con etiquetas, también necesita el permiso `glue:GetConnection` para llamar a las API de etiquetado en las conexiones y el permiso `glue:GetDatabase` para llamar a las API de etiquetado en las bases de datos.

Solicitud

- `ResourceArn` – Obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 10 240 bytes de largo como máximo, que coincide con el Custom string pattern #22.

  ARN del recurso de AWS Glue al que se van a agregar las etiquetas. Para obtener más información acerca de los ARN de recursos de AWS Glue, consulte Patrones de cadena de ARN de AWS Glue.
- `TagsToAdd` – Obligatorio: matriz de mapas de pares clave-valor, con 50 pares como máximo.

  Cada clave es una cadena UTF-8 con una longitud de entre 1 y 128 bytes.

  Cada valor es una cadena UTF-8 que no tiene más de 256 bytes de largo.

  Etiquetas que se van a añadir a este recurso.

Respuesta

- Sin parámetros de respuesta.

Errores

- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `EntityNotFoundException`

Acción UntagResource (Python: untag_resource)

Elimina etiquetas de un recurso.

Solicitud

- `ResourceArn` – Obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 10 240 bytes de largo como máximo, que coincide con el Custom string pattern #22.

  El nombre de recurso de Amazon (ARN) del recurso del que se van a eliminar las etiquetas.
- `TagsToRemove` – Obligatorio: matriz de cadenas UTF-8, con 50 cadenas como máximo.

  Etiquetas que se van a quitar de este recurso.

Respuesta

- Sin parámetros de respuesta.

Errores

- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `EntityNotFoundException`

Acción GetTags (Python: get_tags)

Recupera una lista de etiquetas asociadas a un recurso.

Solicitud

- `ResourceArn` – Obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 10 240 bytes de largo como máximo, que coincide con el Custom string pattern #22.

  El nombre de recurso de Amazon (ARN) del recurso del que se van a recuperar etiquetas.

Respuesta

- `Tags`: matriz de mapas de pares de clave-valor, con 50 pares como máximo.

  Cada clave es una cadena UTF-8 con una longitud de entre 1 y 128 bytes.

  Cada valor es una cadena UTF-8 que no tiene más de 256 bytes de largo.

  Etiquetas solicitadas

Errores

- `InvalidInputException`
- `InternalServiceException`
- `OperationTimeoutException`
- `EntityNotFoundException`

Tipos de datos comunes

Los tipos de datos comunes describen diversos tipos de datos comunes en AWS Glue.

Estructura de etiquetas

El `Tag` objeto representa una etiqueta que se puede asignar a un AWS recurso. Cada etiqueta está formada por una clave y un valor opcional, ambos definidos por el usuario.

Para obtener más información sobre las etiquetas y el control del acceso a los recursos AWS Glue, consulte AWS Etiquetar AWS Glue y especificar los ARN de los AWS Glue recursos en la guía para desarrolladores.

Campos

- `key`: cadena UTF-8, con 1 byte de largo como mínimo y 128 bytes de largo como máximo.

  La clave de la etiqueta. La clave es necesaria para crear una etiqueta en un objeto. La clave distingue entre mayúsculas y minúsculas y no debe contener el prefijo aws.
- `value`: cadena UTF-8 de 256 bytes de largo como máximo.

  El valor de la etiqueta. El valor es opcional para crear una etiqueta en un objeto. El valor distingue entre mayúsculas y minúsculas y no debe contener el prefijo aws.

DecimalNumber estructura

Contiene un valor numérico en formato decimal.

Campos

- `UnscaledValue` – Obligatorio: Blob.

  El valor numérico sin escala.
- `Scale` – Obligatorio: número (entero).

  Escala que determina la ubicación de punto decimal en el valor sin escala.

ErrorDetail estructura

Contiene detalles sobre un error.

Campos

- `ErrorCode`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El código asociado a este error.

- `ErrorMessage`: cadena de descripción de un máximo de 2048 bytes de largo, que coincide con el URI address multi-line string pattern.

  Mensaje que describe el error.

PropertyPredicate estructura

Define el predicado de una propiedad.

Campos

- `Key`: cadena de valor, de 1024 bytes de largo como máximo.

  La clave de la propiedad.
- `Value`: cadena de valor, de 1024 bytes de largo como máximo.

  El valor de la propiedad.
- `Comparator`: cadena UTF-8 (valores válidos: `EQUALS` | `GREATER_THAN` | `LESS_THAN` | `GREATER_THAN_EQUALS` | `LESS_THAN_EQUALS`).

  El comparador utilizado para comparar esta propiedad con otras

ResourceUri estructura

Las URI para los recursos de función.

Campos

- `ResourceType`: cadena UTF-8 (valores válidos: `JAR` | `FILE` | `ARCHIVE`).

  El tipo de recurso.
- `Uri`: identificador uniforme de recursos (uri), con 1 byte de largo como mínimo y 1024 bytes de largo como máximo, que coincide con el URI address multi-line string pattern.

  El URI para obtener acceso al recurso.

ColumnStatistics estructura

Representa las estadísticas de nivel de columna generadas para una tabla o partición.

Campos

- `ColumnName`: obligatorio: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  Nombre de la columna a la que pertenecen las estadísticas.

- `ColumnType` – Obligatorio: nombre de tipo, de 20 000 bytes de largo como máximo, que coincide con Single-line string pattern.

  El tipo de datos de la columna.

- `AnalyzedTime` – Obligatorio: marca temporal.

  Marca temporal del momento en que se generaron las estadísticas de columna.

- `StatisticsData`: obligatorio: objeto ColumnStatisticsData.

  Objeto `ColumnStatisticData` que contiene los valores de datos estadísticos.

ColumnStatisticsError estructura

Encapsula un objeto `ColumnStatistics` que presentó error y el motivo del error.

Campos

- `ColumnStatistics`: un objeto ColumnStatistics.

  Las `ColumnStatistics` de la columna.

- `Error`: un objeto ErrorDetail.

  Un mensaje de error con el motivo del error de una operación.

ColumnError estructura

Encapsula un nombre columna que presentó error y el motivo del error.

Campos

- `ColumnName`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

El nombre de la columna que presentó error.

- `Error`: un objeto ErrorDetail.

  Un mensaje de error con el motivo del error de una operación.

ColumnStatisticsData estructura

Contiene los tipos individuales de datos estadísticos de columna. Solo se debe establecer un objeto de datos e indicar mediante el atributo `Type`.

Campos

- `Type` – Obligatorio: cadena UTF-8 (valores válidos: `BOOLEAN` | `DATE` | `DECIMAL` | `DOUBLE` | `LONG` | `STRING` | `BINARY`).

  Los datos estadísticos del tipo de columna.
- `BooleanColumnStatisticsData`: un objeto BooleanColumnStatisticsData.

  Datos estadísticos de columnas booleanas.
- `DateColumnStatisticsData`: un objeto DateColumnStatisticsData.

  Datos estadísticos de columnas de fecha.
- `DecimalColumnStatisticsData`: un objeto DecimalColumnStatisticsData.

  Datos estadísticos de columnas decimales. UnscaledValues Dentro hay objetos binarios codificados en Base64 que almacenan representaciones en mayodiano, es decir, complementarias de dos, del valor no escalado del decimal.
- `DoubleColumnStatisticsData`: un objeto DoubleColumnStatisticsData.

  Datos estadísticos de doble columna.
- `LongColumnStatisticsData`: un objeto LongColumnStatisticsData.

  Datos estadísticos de columna larga.
- `StringColumnStatisticsData`: un objeto StringColumnStatisticsData.

  Datos estadísticos de columnas de cadena.
- `BinaryColumnStatisticsData`: un objeto BinaryColumnStatisticsData.

Datos estadísticos de columnas binarias.

BooleanColumnStatisticsData estructura

Define las estadísticas de columna soportadas para las columnas de datos booleanos.

Campos

- `NumberOfTrues` – Obligatorio: número (largo), cero como máximo.

  El número de valores verdaderos en la columna.
- `NumberOfFalses` – Obligatorio: número (largo), cero como máximo.

  El número de valores falsos en la columna.
- `NumberOfNulls` – Obligatorio: número (largo), cero como máximo.

  El número de valores nulos en la columna.

DateColumnStatisticsData estructura

Define las estadísticas de columna soportadas para las columnas de datos de marca temporal.

Campos

- `MinimumValue`: marca temporal.

  El valor más bajo de la columna.
- `MaximumValue`: marca temporal.

  El valor más alto de la columna.
- `NumberOfNulls` – Obligatorio: número (largo), cero como máximo.

  El número de valores nulos en la columna.
- `NumberOfDistinctValues` – Obligatorio: número (largo), cero como máximo.

  El número de valores distintos de una columna.

DecimalColumnStatisticsData estructura

Define las estadísticas de columna soportadas para las columnas de datos de números con coma fija.

Campos

- `MinimumValue`: un objeto DecimalNumber.

  El valor más bajo de la columna.
- `MaximumValue`: un objeto DecimalNumber.

  El valor más alto de la columna.
- `NumberOfNulls` – Obligatorio: número (largo), cero como máximo.

  El número de valores nulos en la columna.
- `NumberOfDistinctValues` – Obligatorio: número (largo), cero como máximo.

  El número de valores distintos de una columna.

DoubleColumnStatisticsData estructura

Define las estadísticas de columna soportadas para las columnas de datos de números con coma flotante.

Campos

- `MinimumValue`: número (doble).

  El valor más bajo de la columna.
- `MaximumValue`: número (doble).

  El valor más alto de la columna.
- `NumberOfNulls` – Obligatorio: número (largo), cero como máximo.

  El número de valores nulos en la columna.
- `NumberOfDistinctValues` – Obligatorio: número (largo), cero como máximo.

  El número de valores distintos de una columna.

LongColumnStatisticsData estructura

Define las estadísticas de columna soportadas para las columnas de datos enteros.

Campos

- `MinimumValue`: número (largo).

  El valor más bajo de la columna.
- `MaximumValue`: número (largo).

  El valor más alto de la columna.
- `NumberOfNulls` – Obligatorio: número (largo), cero como máximo.

  El número de valores nulos en la columna.
- `NumberOfDistinctValues` – Obligatorio: número (largo), cero como máximo.

  El número de valores distintos de una columna.

StringColumnStatisticsData estructura

Define estadísticas de columna admitidas para valores de datos de secuencia de caracteres.

Campos

- `MaximumLength` – Obligatorio: número (largo), cero como máximo.

  El tamaño de la cadena más larga de la columna.
- `AverageLength` – Obligatorio: número (doble), cero como máximo.

  La longitud media de la cadena en la columna.
- `NumberOfNulls` – Obligatorio: número (largo), cero como máximo.

  El número de valores nulos en la columna.
- `NumberOfDistinctValues` – Obligatorio: número (largo), cero como máximo.

  El número de valores distintos de una columna.

BinaryColumnStatisticsData estructura

Define estadísticas de columna soportadas para valores de datos de secuencia de bits.

Campos

- `MaximumLength` – Obligatorio: número (largo), cero como máximo.

  El tamaño de la secuencia de bits más larga de la columna.
- `AverageLength` – Obligatorio: número (doble), cero como máximo.

  La longitud media de secuencia de bits en la columna.
- `NumberOfNulls` – Obligatorio: número (largo), cero como máximo.

  El número de valores nulos en la columna.

Patrones de cadena

La API utiliza las siguientes expresiones regulares para definir qué es un contenido válido para diversos miembros y parámetros de cadena:

- Patrón de cadena de línea única: "`[\u0020-\uD7FF\uE000-\uFFFD\uD800\uDC00-\uDBFF\uDFFF\t]*`"
- Patrón de cadena de varias líneas de la dirección URI: "`[\u0020-\uD7FF\uE000-\uFFFD\uD800\uDC00-\uDBFF\uDFFF\r\n\t]*`"
- Patrón de cadena Grok Logstash: "`[\u0020-\uD7FF\uE000-\uFFFD\uD800\uDC00-\uDBFF\uDFFF\r\t]*`"
- Patrón de cadena del identificador: "`[A-Za-z_][A-Za-z0-9_]*`"
- Patrón de cadena de ARN de AWS IAM: "`arn:aws:iam::\d{12}:role/.*`"
- Patrón de cadena de versión: "`^[a-zA-Z0-9-_]+$`"
- Patrón de cadena de grupo de registros: "`[\.\-_/#A-Za-z0-9]+`"
- Patrón de cadena de flujo de registro: "`[^:*]*`"
- Patrón de cadena personalizado n.º 10: “`[^\r\n]`”
- Patrón de cadena personalizado n.º 11: “`^arn:aws(-(cn|us-gov|iso(-[bef])?))?:secretsmanager:.*$`”

- Patrón de cadena personalizado n.º 12: “^(https?)://[-a-zA-Z0-9+@#/%?=~_|!:,.;]*[-a-zA-Z0-9+@#/%=~_|]”
- Patrón de cadena personalizado n.º 13: “\S+”
- Patrón de cadena personalizado n.º 14: “^(https?):\/\/[^\s/$.?#].[^\s]*$”
- Patrón de cadena personalizado n.º 15: “^subnet-[a-z0-9]+$”
- Patrón de cadena personalizado n.º 16: “[\p{L}\p{N}\p{P}]*”
- Patrón de cadena personalizado n.º 17: “[a-f0-9]{8}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{12}”
- Patrón de cadena personalizado n.º 18: “[a-zA-Z0-9-_$#.]+”
- Patrón de cadena personalizado n.º 19: “^\w+\.\w+\.\w+$”
- Patrón de cadena personalizado n.º 20: “^\w+\.\w+$”
- Patrón de cadena personalizado n.º 21: “^([2-3]|3[.]9)$”
- Patrón de cadena personalizado n.º 22: “arn:(aws|aws-us-gov|aws-cn):glue:.*”
- Patrón de cadena personalizado n.º 23: “(^arn:aws:iam::\w{12}:root)”
- Patrón de cadena personalizado n.º 24: “^arn:aws(-(cn|us-gov|iso(-[bef])?))?:iam::[0-9]{12}:role/.+”
- Patrón de cadena personalizado n.º 25: “arn:aws:kms:.*”
- Patrón de cadena personalizado n.º 26: “arn:aws[^:]*:iam::[0-9]*:role/.+”
- Patrón de cadena personalizado n.º 27: “[\.\-_A-Za-z0-9]+”
- Patrón de cadena personalizado n.º 28: “^s3://([^/]+)/([^/]+/)*([^/]+)$”
- Patrón de cadena personalizado n.º 29: “.*”
- Patrón de cadena personalizado n.º 30: “^(Sun|Mon|Tue|Wed|Thu|Fri|Sat):([01]?[0-9]|2[0-3])$”
- Patrón de cadena personalizado n.º 31: “[a-zA-Z0-9_.-]+”
- Patrón de cadena personalizado n.º 32: “.*\S.*”
- Patrón de cadena personalizado n.º 33: “[a-zA-Z0-9-=._/@]+”
- Patrón de cadena personalizado n.º 34: “[1-9][0-9]*|[1-9][0-9]*-[1-9][0-9]*”
- Patrón de cadena personalizado n.º 35: “[\s\S]*”
- Patrón de cadena personalizado N.º 36: “([\u0020-\uD7FF\uE000-\uFFFD\uD800\uDC00-\uDBFF\uDFFF]|[^\S\r\n"'= ;])*”
- Patrón de cadena personalizado N.º 37: “[*A-Za-z0-9_-]*”

- Patrón de cadena personalizado N.º 38: «([\u0020-\u007E\r\s\n])*»
- Patrón de cadena personalizado N.º 39: «[A-Za-z0-9_-]*»
- Patrón de cadena personalizado N.º 40: «([\u0020-\uD7FF\uE000-\uFFFD\uD800\uDC00-\uDBFF\uDFFF]|[^\S\r\n"'])*»
- Patrón de cadena personalizado N.º 41: «([\u0020-\uD7FF\uE000-\uFFFD\uD800\uDC00-\uDBFF\uDFFF]|[^\S\r\n])*»
- Patrón de cadena personalizado N.º 42: «([\u0020-\uD7FF\uE000-\uFFFD\uD800\uDC00-\uDBFF\uDFFF\s])*»
- Patrón de cadena personalizado #43 — "([\u0020-\uD7FF\uE000-\uFFFD\uD800\uDC00-\uDBFF\uDFFF]|[^\r\n])*»

Excepciones

En esta sección, se describen las excepciones de AWS Glue que puede utilizar para encontrar el origen de los problemas y solucionarlos. Para obtener más información sobre las cadenas y códigos de error HTTP de las excepciones relacionadas con el machine learning, consulte the section called “Excepciones de machine learning en AWS Glue”.

Estructura AccessDeniedException

El acceso a un recurso se ha denegado.

Campos

- `Message`: cadena UTF-8.

  Mensaje que describe el problema.

Estructura de AlreadyExistsException

Un recurso que se va a crear o añadir ya existe.

Campos

- `Message`: cadena UTF-8.

  Mensaje que describe el problema.

Estructura ConcurrentModificationException

Dos procesos intentan modificar un recurso a la vez.

Campos

- `Message`: cadena UTF-8.

  Mensaje que describe el problema.

Estructura ConcurrentRunsExceededException

Demasiados trabajos se ejecutan al mismo tiempo.

Campos

- `Message`: cadena UTF-8.

  Mensaje que describe el problema.

Estructura CrawlerNotRunningException

El rastreador especificado no está ejecutándose.

Campos

- `Message`: cadena UTF-8.

  Mensaje que describe el problema.

Estructura CrawlerRunningException

La operación no se puede realizar porque el rastreador ya se está ejecutando.

Campos

- `Message`: cadena UTF-8.

  Mensaje que describe el problema.

Estructura CrawlerStoppingException

El rastreador especificado se está parando.

Campos

- `Message`: cadena UTF-8.

  Mensaje que describe el problema.

Estructura EntityNotFoundException

Una entidad especificada no existe.

Campos

- `Message`: cadena UTF-8.

  Mensaje que describe el problema.
- `FromFederationSource`: booleano.

  Indica si la excepción se refiere o no a un origen federado.

Estructura de FederationSourceException

Error en un origen de federación.

Campos

- `FederationSourceErrorCode`: Cadena UTF-8 (valores válidos: `AccessDeniedException` | `EntityNotFoundException` | `InvalidCredentialsException` | `InvalidInputException` | `InvalidResponseException` | `OperationTimeoutException` | `OperationNotSupportedException` | `InternalServiceException` | `PartialFailureException` | `ThrottlingException`).

  El código de error del problema.
- `Message`: cadena UTF-8.

  Mensaje que describe el problema.

Estructura de FederationSourceRetryableException

Se produjo un error en un origen de federación, pero es posible que se vuelva a intentar la operación.

Campos

- `Message`: cadena UTF-8.

  Mensaje que describe el problema.

Estructura GlueEncryptionException

Una operación de cifrado ha devuelto un error.

Campos

- `Message`: cadena UTF-8.

  Mensaje que describe el problema.

Estructura IdempotentParameterMismatchException

El mismo identificador único se asoció a dos registros diferentes.

Campos

- `Message`: cadena UTF-8.

  Mensaje que describe el problema.

Estructura IllegalWorkflowStateException

El flujo de trabajo está en un estado no válido para realizar una operación solicitada.

Campos

- `Message`: cadena UTF-8.

  Mensaje que describe el problema.

Estructura InternalServiceException

Se ha producido un error de servicio interno.

Campos

- `Message`: cadena UTF-8.

  Mensaje que describe el problema.

Estructura InvalidExecutionEngineException

Se ha especificado un motor de ejecución desconocido o no válido.

Campos

- `message`: cadena UTF-8.

  Mensaje que describe el problema.

Estructura InvalidInputException

La información de entrada proporcionada no es válida.

Campos

- `Message`: cadena UTF-8.

  Mensaje que describe el problema.
- `FromFederationSource`: booleano.

  Indica si la excepción se refiere o no a un origen federado.

Estructura InvalidStateException

Un error que indica que los datos se encuentran en un estado no válido.

Campos

- `Message`: cadena UTF-8.

Mensaje que describe el problema.

Estructura InvalidTaskStatusTransitionException

Error en la transición correcta de una tarea a la siguiente.

Campos

- `message`: cadena UTF-8.

  Mensaje que describe el problema.

Estructura JobDefinitionErrorException

Una definición de trabajo no es válida.

Campos

- `message`: cadena UTF-8.

  Mensaje que describe el problema.

Estructura JobRunInTerminalStateException

El estado terminal de una ejecución de flujo de trabajo señala un error.

Campos

- `message`: cadena UTF-8.

  Mensaje que describe el problema.

Estructura JobRunInvalidStateTransitionException

Una ejecución de flujo de trabajo ha encontrado una transición no válida del estado de origen al de destino.

Campos

- `jobRunId`: cadena UTF-8, con 1 byte de largo como mínimo y 255 bytes de largo como máximo, que coincide con el Single-line string pattern.

  El ID de la ejecución de trabajo en cuestión.

- `message`: cadena UTF-8.

  Mensaje que describe el problema.

- `sourceState`: Cadena UTF-8 (valores válidos: `STARTING` | `RUNNING` | `STOPPING` | `STOPPED` | `SUCCEEDED` | `FAILED` | `TIMEOUT` | `ERROR` | `WAITING` | `EXPIRED`).

  El estado de origen.

- `targetState`: Cadena UTF-8 (valores válidos: `STARTING` | `RUNNING` | `STOPPING` | `STOPPED` | `SUCCEEDED` | `FAILED` | `TIMEOUT` | `ERROR` | `WAITING` | `EXPIRED`).

  El estado de destino.

Estructura JobRunNotInTerminalStateException

Una ejecución de flujo de trabajo no se encuentra en estado terminal.

Campos

- `message`: cadena UTF-8.

  Mensaje que describe el problema.

Estructura LateRunnerException

Un ejecutor de flujo de trabajo se ha atrasado.

Campos

- `Message`: cadena UTF-8.

  Mensaje que describe el problema.

Estructura NoScheduleException

No hay ningún programa aplicable.

Campos

- `Message`: cadena UTF-8.

  Mensaje que describe el problema.

Estructura OperationTimeoutException

La operación ha agotado el tiempo de espera.

Campos

- `Message`: cadena UTF-8.

  Mensaje que describe el problema.

Estructura ResourceNotReadyException

Un recurso no estaba listo para una transacción.

Campos

- `Message`: cadena UTF-8.

  Mensaje que describe el problema.

Estructura ResourceNumberLimitExceededException

Un límite numérico de los recursos se ha superado.

Campos

- `Message`: cadena UTF-8.

  Mensaje que describe el problema.

Estructura SchedulerNotRunningException

El programador especificado no está ejecutándose.

Campos

- `Message`: cadena UTF-8.

  Mensaje que describe el problema.

Estructura SchedulerRunningException

El programador especificado ya está ejecutándose.

Campos

- `Message`: cadena UTF-8.

  Mensaje que describe el problema.

Estructura SchedulerTransitioningException

El programador especificado se encuentra en transición.

Campos

- `Message`: cadena UTF-8.

  Mensaje que describe el problema.

Estructura UnrecognizedRunnerException

No se ha reconocido al ejecutor del flujo de trabajo.

Campos

- `Message`: cadena UTF-8.

  Mensaje que describe el problema.

Estructura ValidationException

No se ha podido validar a un valor.

Campos

- `Message`: cadena UTF-8.

  Mensaje que describe el problema.

Estructura VersionMismatchException

Se ha producido un conflicto de versiones.

Campos

- `Message`: cadena UTF-8.

  Mensaje que describe el problema.

AWS Glue Ejemplos de código de API con AWS SDK

Los siguientes ejemplos de código muestran cómo usarlo AWS Glue con un kit de desarrollo de AWS software (SDK).

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las funciones de servicio individuales, es posible ver las acciones en contexto en los escenarios relacionados y en los ejemplos entre servicios.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica llamando a varias funciones dentro del mismo servicio.

Para obtener una lista completa de las guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de este servicio con un AWS SDK. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Introducción

Hola AWS Glue

En los siguientes ejemplos de código se muestra cómo empezar a utilizar AWS Glue.

.NET

AWS SDK for .NET

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
namespace GlueActions;

public class HelloGlue
{
    private static ILogger logger = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for AWS Glue.
```

```
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
 LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonGlue>()
                .AddTransient<GlueWrapper>()
            )
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<HelloGlue>();
        var glueClient = host.Services.GetRequiredService<IAmazonGlue>();

        var request = new ListJobsRequest();

        var jobNames = new List<string>();

        do
        {
            var response = await glueClient.ListJobsAsync(request);
            jobNames.AddRange(response.JobNames);
            request.NextToken = response.NextToken;
        }
        while (request.NextToken is not null);

        Console.Clear();
        Console.WriteLine("Hello, Glue. Let's list your existing Glue Jobs:");
        if (jobNames.Count == 0)
        {
            Console.WriteLine("You don't have any AWS Glue jobs.");
        }
        else
        {
            jobNames.ForEach(Console.WriteLine);
        }
    }
}
```

- Para obtener más información sobre la API, consulta ListJobsla Referencia AWS SDK for .NET de la API.

C++

SDK para C++

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Código para el MakeLists archivo CMake C.txt.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS glue)

# Set this project's name.
project("hello_glue")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
```

```
    # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

    # set(BIN_SUB_DIR "/Debug") # if you are building from the command line you
 may need to uncomment this
                                # and set the proper subdirectory to the
 executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_glue.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

Código del archivo de origen hello_glue.cpp.

```
#include <aws/core/Aws.h>
#include <aws/glue/GlueClient.h>
#include <aws/glue/model/ListJobsRequest.h>
#include <iostream>

/*
 *  A "Hello Glue" starter application which initializes an AWS Glue client and
 lists the
 *  AWS Glue job definitions.
 *
 *  main function
 *
 *  Usage: 'hello_glue'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
```

```
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::Glue::GlueClient glueClient(clientConfig);

        std::vector<Aws::String> jobs;

        Aws::String nextToken;  // Used for pagination.
        do {
            Aws::Glue::Model::ListJobsRequest listJobsRequest;
            if (!nextToken.empty()) {
                listJobsRequest.SetNextToken(nextToken);
            }

            Aws::Glue::Model::ListJobsOutcome listRunsOutcome =
 glueClient.ListJobs(
                    listJobsRequest);

            if (listRunsOutcome.IsSuccess()) {
                const std::vector<Aws::String> &jobNames =
 listRunsOutcome.GetResult().GetJobNames();
                jobs.insert(jobs.end(), jobNames.begin(), jobNames.end());

                nextToken = listRunsOutcome.GetResult().GetNextToken();
            } else {
                std::cerr << "Error listing jobs. "
                          << listRunsOutcome.GetError().GetMessage()
                          << std::endl;
                result = 1;
                break;
            }
        } while (!nextToken.empty());

        std::cout << "Your account has " << jobs.size() << " jobs."
                  << std::endl;
        for (size_t i = 0; i < jobs.size(); ++i) {
            std::cout << "   " << i + 1 << ". " << jobs[i] << std::endl;
        }
    }
    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}
```

- Para obtener más información sobre la API, consulte la Referencia de ListJobsla AWS SDK for C++ API.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
package com.example.glue;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.ListJobsRequest;
import software.amazon.awssdk.services.glue.model.ListJobsResponse;
import java.util.List;

public class HelloGlue {
    public static void main(String[] args) {
        GlueClient glueClient = GlueClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listJobs(glueClient);
    }

    public static void listJobs(GlueClient glueClient) {
        ListJobsRequest request = ListJobsRequest.builder()
                .maxResults(10)
                .build();
        ListJobsResponse response = glueClient.listJobs(request);
        List<String> jobList = response.jobNames();
        jobList.forEach(job -> {
            System.out.println("Job Name: " + job);
        });
```

```
    }
}
```

- Para obtener más información sobre la API, consulta ListJobsla Referencia AWS SDK for Java 2.x de la API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import { ListJobsCommand, GlueClient } from "@aws-sdk/client-glue";

const client = new GlueClient({});

export const main = async () => {
  const command = new ListJobsCommand({});

  const { JobNames } = await client.send(command);
  const formattedJobNames = JobNames.join("\n");
  console.log("Job names: ");
  console.log(formattedJobNames);
  return JobNames;
};
```

- Para obtener más información sobre la API, consulta ListJobsla Referencia AWS SDK for JavaScript de la API.

Rust

SDK para Rust

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    let mut list_jobs = glue.list_jobs().into_paginator().send();
    while let Some(list_jobs_output) = list_jobs.next().await {
        match list_jobs_output {
            Ok(list_jobs) => {
                let names = list_jobs.job_names();
                info!(?names, "Found these jobs")
            }
            Err(err) => return Err(GlueMvpError::from_glue_sdk(err)),
        }
    }
```

- Para obtener más información sobre la API, consulta ListJobsla referencia sobre la API de AWS SDK para Rust.

Ejemplos de código

- Acciones para AWS Glue usar los AWS SDK
  - Úselo CreateCrawler con un AWS SDK o CLI
  - Úselo CreateJob con un AWS SDK o CLI
  - Úselo DeleteCrawler con un AWS SDK o CLI
  - Úselo DeleteDatabase con un AWS SDK o CLI
  - Úselo DeleteJob con un AWS SDK o CLI
  - Úselo DeleteTable con un AWS SDK o CLI
  - Úselo GetCrawler con un AWS SDK o CLI
  - Úselo GetDatabase con un AWS SDK o CLI
  - Úselo GetDatabases con un AWS SDK o CLI

- Úselo GetJob con un AWS SDK o CLI
- Úselo GetJobRun con un AWS SDK o CLI
- Úselo GetJobRuns con un AWS SDK o CLI
- Úselo GetTables con un AWS SDK o CLI
- Úselo ListJobs con un AWS SDK o CLI
- Úselo StartCrawler con un AWS SDK o CLI
- Úselo StartJobRun con un AWS SDK o CLI

- Escenarios de AWS Glue uso de AWS los SDK
  - Comience a ejecutar AWS Glue rastreadores y tareas con un AWS SDK

Acciones para AWS Glue usar los AWS SDK

Los siguientes ejemplos de código muestran cómo realizar AWS Glue acciones individuales con los AWS SDK. Estos extractos se denominan AWS Glue API y son fragmentos de código de programas más grandes que deben ejecutarse en su contexto. Cada ejemplo incluye un enlace a GitHub, donde puede encontrar instrucciones para configurar y ejecutar el código.

Los siguientes ejemplos incluyen solo las acciones que se utilizan con mayor frecuencia. Para ver una lista completa, consulte la Referencia de la API de AWS Glue.

Ejemplos

- Úselo CreateCrawler con un AWS SDK o CLI
- Úselo CreateJob con un AWS SDK o CLI
- Úselo DeleteCrawler con un AWS SDK o CLI
- Úselo DeleteDatabase con un AWS SDK o CLI
- Úselo DeleteJob con un AWS SDK o CLI
- Úselo DeleteTable con un AWS SDK o CLI
- Úselo GetCrawler con un AWS SDK o CLI
- Úselo GetDatabase con un AWS SDK o CLI
- Úselo GetDatabases con un AWS SDK o CLI
- Úselo GetJob con un AWS SDK o CLI
- Úselo GetJobRun con un AWS SDK o CLI
- Úselo GetJobRuns con un AWS SDK o CLI

- Úselo GetTables con un AWS SDK o CLI
- Úselo ListJobs con un AWS SDK o CLI
- Úselo StartCrawler con un AWS SDK o CLI
- Úselo StartJobRun con un AWS SDK o CLI

Úselo **`CreateCrawler`** con un AWS SDK o CLI

En los siguientes ejemplos de código, se muestra cómo utilizar `CreateCrawler`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- Comenzar a ejecutar rastreadores y trabajos

.NET

AWS SDK for .NET

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    /// <summary>
    /// Create an AWS Glue crawler.
    /// </summary>
    /// <param name="crawlerName">The name for the crawler.</param>
    /// <param name="crawlerDescription">A description of the crawler.</param>
    /// <param name="role">The AWS Identity and Access Management (IAM) role to
    /// be assumed by the crawler.</param>
    /// <param name="schedule">The schedule on which the crawler will be
executed.</param>
    /// <param name="s3Path">The path to the Amazon Simple Storage Service
(Amazon S3)
    /// bucket where the Python script has been stored.</param>
    /// <param name="dbName">The name to use for the database that will be
    /// created by the crawler.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
```

```
    public async Task<bool> CreateCrawlerAsync(
        string crawlerName,
        string crawlerDescription,
        string role,
        string schedule,
        string s3Path,
        string dbName)
    {
        var s3Target = new S3Target
        {
            Path = s3Path,
        };

        var targetList = new List<S3Target>
        {
            s3Target,
        };

        var targets = new CrawlerTargets
        {
            S3Targets = targetList,
        };

        var crawlerRequest = new CreateCrawlerRequest
        {
            DatabaseName = dbName,
            Name = crawlerName,
            Description = crawlerDescription,
            Targets = targets,
            Role = role,
            Schedule = schedule,
        };

        var response = await _amazonGlue.CreateCrawlerAsync(crawlerRequest);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- Para obtener más información sobre la API, consulta CreateCrawlerla Referencia AWS SDK for .NET de la API.

C++

SDK para C++

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::S3Target s3Target;
        s3Target.SetPath("s3://crawler-public-us-east-1/flight/2016/csv");
        Aws::Glue::Model::CrawlerTargets crawlerTargets;
        crawlerTargets.AddS3Targets(s3Target);

        Aws::Glue::Model::CreateCrawlerRequest request;
        request.SetTargets(crawlerTargets);
        request.SetName(CRAWLER_NAME);
        request.SetDatabaseName(CRAWLER_DATABASE_NAME);
        request.SetTablePrefix(CRAWLER_DATABASE_PREFIX);
        request.SetRole(roleArn);

        Aws::Glue::Model::CreateCrawlerOutcome outcome =
client.CreateCrawler(request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully created the crawler." << std::endl;
        }
        else {
            std::cerr << "Error creating a crawler. " <<
outcome.GetError().GetMessage()
                      << std::endl;
            deleteAssets("", CRAWLER_DATABASE_NAME, "", bucketName,
clientConfig);
            return false;
```

```
        }
```

- Para obtener más información sobre la API, consulta CreateCrawlerla Referencia AWS SDK for C++ de la API.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.CreateCrawlerRequest;
import software.amazon.awssdk.services.glue.model.CrawlerTargets;
import software.amazon.awssdk.services.glue.model.GlueException;
import software.amazon.awssdk.services.glue.model.S3Target;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateCrawler {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <IAM> <s3Path> <cron> <dbName> <crawlerName>
```

```
                Where:
                    IAM - The ARN of the IAM role that has AWS Glue and S3
permissions.\s
                    s3Path - The Amazon Simple Storage Service (Amazon S3) target
that contains data (for example, CSV data).
                    cron - A cron expression used to specify the schedule  (i.e.,
cron(15 12 * * ? *).
                    dbName - The database name.\s
                    crawlerName - The name of the crawler.\s
                """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String iam = args[0];
        String s3Path = args[1];
        String cron = args[2];
        String dbName = args[3];
        String crawlerName = args[4];
        Region region = Region.US_EAST_1;
        GlueClient glueClient = GlueClient.builder()
                .region(region)
                .build();

        createGlueCrawler(glueClient, iam, s3Path, cron, dbName, crawlerName);
        glueClient.close();
    }

    public static void createGlueCrawler(GlueClient glueClient,
            String iam,
            String s3Path,
            String cron,
            String dbName,
            String crawlerName) {

        try {
            S3Target s3Target = S3Target.builder()
                    .path(s3Path)
                    .build();

            // Add the S3Target to a list.
            List<S3Target> targetList = new ArrayList<>();
```

```
            targetList.add(s3Target);

            CrawlerTargets targets = CrawlerTargets.builder()
                    .s3Targets(targetList)
                    .build();

            CreateCrawlerRequest crawlerRequest = CreateCrawlerRequest.builder()
                    .databaseName(dbName)
                    .name(crawlerName)
                    .description("Created by the AWS Glue Java API")
                    .targets(targets)
                    .role(iam)
                    .schedule(cron)
                    .build();

            glueClient.createCrawler(crawlerRequest);
            System.out.println(crawlerName + " was successfully created");

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obtener más información sobre la API, consulta CreateCrawlerla Referencia AWS SDK for Java 2.x de la API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
const createCrawler = (name, role, dbName, tablePrefix, s3TargetPath) => {
  const client = new GlueClient({});
```

```
  const command = new CreateCrawlerCommand({
    Name: name,
    Role: role,
    DatabaseName: dbName,
    TablePrefix: tablePrefix,
    Targets: {
      S3Targets: [{ Path: s3TargetPath }],
    },
  });

  return client.send(command);
};
```

- Para obtener más información sobre la API, consulta CreateCrawlerla Referencia AWS SDK for JavaScript de la API.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun createGlueCrawler(
    iam: String?,
    s3Path: String?,
    cron: String?,
    dbName: String?,
    crawlerName: String,
) {
    val s3Target =
        S3Target {
            path = s3Path
        }

    // Add the S3Target to a list.
```

```
    val targetList = mutableListOf<S3Target>()
    targetList.add(s3Target)

    val targetOb =
        CrawlerTargets {
            s3Targets = targetList
        }

    val request =
        CreateCrawlerRequest {
            databaseName = dbName
            name = crawlerName
            description = "Created by the AWS Glue Kotlin API"
            targets = targetOb
            role = iam
            schedule = cron
        }

    GlueClient { region = "us-west-2" }.use { glueClient ->
        glueClient.createCrawler(request)
        println("$crawlerName was successfully created")
    }
}
```

- Para obtener más información sobre la API, consulta CreateCrawlerla referencia sobre el AWS SDK para la API de Kotlin.

PHP

SDK para PHP

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
        $crawlerName = "example-crawler-test-" . $uniqid;

        $role = $iamService->getRole("AWSGlueServiceRole-DocExample");
```

```
        $path = 's3://crawler-public-us-east-1/flight/2016/csv';
        $glueService->createCrawler($crawlerName, $role['Role']['Arn'],
$databaseName, $path);

    public function createCrawler($crawlerName, $role, $databaseName, $path):
Result
    {
        return $this->customWaiter(function () use ($crawlerName, $role,
$databaseName, $path) {
            return $this->glueClient->createCrawler([
                'Name' => $crawlerName,
                'Role' => $role,
                'DatabaseName' => $databaseName,
                'Targets' => [
                    'S3Targets' =>
                        [[
                            'Path' => $path,
                        ]]
                ],
            ]);
        });
    }
```

- Para obtener más información sobre la API, consulta CreateCrawlerla Referencia AWS SDK for PHP de la API.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""
```

```
    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def create_crawler(self, name, role_arn, db_name, db_prefix, s3_target):
        """
        Creates a crawler that can crawl the specified target and populate a
        database in your AWS Glue Data Catalog with metadata that describes the
data
        in the target.

        :param name: The name of the crawler.
        :param role_arn: The Amazon Resource Name (ARN) of an AWS Identity and
Access
                         Management (IAM) role that grants permission to let AWS
Glue
                         access the resources it needs.
        :param db_name: The name to give the database that is created by the
crawler.
        :param db_prefix: The prefix to give any database tables that are created
by
                          the crawler.
        :param s3_target: The URL to an S3 bucket that contains data that is
                          the target of the crawler.
        """
        try:
            self.glue_client.create_crawler(
                Name=name,
                Role=role_arn,
                DatabaseName=db_name,
                TablePrefix=db_prefix,
                Targets={"S3Targets": [{"Path": s3_target}]},
            )
        except ClientError as err:
            logger.error(
                "Couldn't create crawler. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obtener más información sobre la API, consulta CreateCrawlerla AWS Referencia de API de SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Creates a new crawler with the specified configuration.
  #
  # @param name [String] The name of the crawler.
  # @param role_arn [String] The ARN of the IAM role to be used by the crawler.
  # @param db_name [String] The name of the database where the crawler stores its
 metadata.
  # @param db_prefix [String] The prefix to be added to the names of tables that
 the crawler creates.
  # @param s3_target [String] The S3 path that the crawler will crawl.
  # @return [void]
  def create_crawler(name, role_arn, db_name, db_prefix, s3_target)
    @glue_client.create_crawler(
      name: name,
```

```
      role: role_arn,
      database_name: db_name,
      targets: {
        s3_targets: [
          {
            path: s3_target
          }
        ]
      }
    )
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not create crawler: \n#{e.message}")
    raise
  end
```

- Para obtener más información sobre la API, consulta CreateCrawlerla Referencia AWS SDK for Ruby de la API.

Rust

SDK para Rust

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
        let create_crawler = glue
            .create_crawler()
            .name(self.crawler())
            .database_name(self.database())
            .role(self.iam_role.expose_secret())
            .targets(
                CrawlerTargets::builder()
                    .s3_targets(S3Target::builder().path(CRAWLER_TARGET).build())
                    .build(),
            )
            .send()
            .await;
```

```
        match create_crawler {
            Err(err) => {
                let glue_err: aws_sdk_glue::Error = err.into();
                match glue_err {
                    aws_sdk_glue::Error::AlreadyExistsException(_) => {
                        info!("Using existing crawler");
                        Ok(())
                    }
                    _ => Err(GlueMvpError::GlueSdk(glue_err)),
                }
            }
            Ok(_) => Ok(()),
        }?;
```

- Para obtener más información sobre la API, consulta CreateCrawlerla referencia sobre la API de AWS SDK para Rust.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de este servicio con un AWS SDK. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`CreateJob`** con un AWS SDK o CLI

En los siguientes ejemplos de código, se muestra cómo utilizar `CreateJob`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- Comenzar a ejecutar rastreadores y trabajos

.NET

AWS SDK for .NET

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    /// <summary>
    /// Create an AWS Glue job.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <param name="roleName">The name of the IAM role to be assumed by
    /// the job.</param>
    /// <param name="description">A description of the job.</param>
    /// <param name="scriptUrl">The URL to the script.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> CreateJobAsync(string dbName, string tableName,
string bucketUrl, string jobName, string roleName, string description, string
scriptUrl)
    {
        var command = new JobCommand
        {
            PythonVersion = "3",
            Name = "glueetl",
            ScriptLocation = scriptUrl,
        };

        var arguments = new Dictionary<string, string>
        {
            { "--input_database", dbName },
            { "--input_table", tableName },
            { "--output_bucket_url", bucketUrl }
        };

        var request = new CreateJobRequest
        {
            Command = command,
            DefaultArguments = arguments,
            Description = description,
            GlueVersion = "3.0",
            Name = jobName,
            NumberOfWorkers = 10,
            Role = roleName,
            WorkerType = "G.1X"
        };

        var response = await _amazonGlue.CreateJobAsync(request);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Para obtener más información sobre la API, consulta CreateJobla Referencia AWS SDK for .NET de la API.

C++

SDK para C++

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::CreateJobRequest request;
        request.SetName(JOB_NAME);
        request.SetRole(roleArn);
        request.SetGlueVersion(GLUE_VERSION);

        Aws::Glue::Model::JobCommand command;
        command.SetName(JOB_COMMAND_NAME);
        command.SetPythonVersion(JOB_PYTHON_VERSION);
        command.SetScriptLocation(
                Aws::String("s3://") + bucketName + "/" + PYTHON_SCRIPT);
        request.SetCommand(command);

        Aws::Glue::Model::CreateJobOutcome outcome = client.CreateJob(request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully created the job." << std::endl;
        }
        else {
```

```
            std::cerr << "Error creating the job. " <<
 outcome.GetError().GetMessage()
                      << std::endl;
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                         clientConfig);
            return false;
        }
```

- Para obtener más información sobre la API, consulta CreateJobla Referencia AWS SDK for C++ de la API.

CLI

AWS CLI

Para crear un trabajo a fin de transformar datos

El siguiente ejemplo de `create-job` crea un trabajo de streaming que ejecuta un script almacenado en S3.

```
aws glue create-job \
    --name my-testing-job \
    --role AWSGlueServiceRoleDefault \
    --command '{ \
        "Name": "gluestreaming", \
        "ScriptLocation": "s3://DOC-EXAMPLE-BUCKET/folder/" \
    }' \
    --region us-east-1 \
    --output json \
    --default-arguments '{ \
        "--job-language":"scala", \
        "--class":"GlueApp" \
    }' \
    --profile my-profile \
    --endpoint https://glue.us-east-1.amazonaws.com
```

Contenidos de `test_script.scala`:

```
import com.amazonaws.services.glue.ChoiceOption
import com.amazonaws.services.glue.GlueContext
import com.amazonaws.services.glue.MappingSpec
```

```
import com.amazonaws.services.glue.ResolveSpec
import com.amazonaws.services.glue.errors.CallSite
import com.amazonaws.services.glue.util.GlueArgParser
import com.amazonaws.services.glue.util.Job
import com.amazonaws.services.glue.util.JsonOptions
import org.apache.spark.SparkContext
import scala.collection.JavaConverters._

object GlueApp {
    def main(sysArgs: Array[String]) {
        val spark: SparkContext = new SparkContext()
        val glueContext: GlueContext = new GlueContext(spark)
        // @params: [JOB_NAME]
        val args = GlueArgParser.getResolvedOptions(sysArgs,
 Seq("JOB_NAME").toArray)
        Job.init(args("JOB_NAME"), glueContext, args.asJava)
        // @type: DataSource
        // @args: [database = "tempdb", table_name = "s3-source",
 transformation_ctx = "datasource0"]
        // @return: datasource0
        // @inputs: []
        val datasource0 = glueContext.getCatalogSource(database = "tempdb",
 tableName = "s3-source", redshiftTmpDir = "", transformationContext =
 "datasource0").getDynamicFrame()
        // @type: ApplyMapping
        // @args: [mapping = [("sensorid", "int", "sensorid", "int"),
 ("currenttemperature", "int", "currenttemperature", "int"), ("status", "string",
 "status", "string")], transformation_ctx = "applymapping1"]
        // @return: applymapping1
        // @inputs: [frame = datasource0]
        val applymapping1 = datasource0.applyMapping(mappings = Seq(("sensorid",
 "int", "sensorid", "int"), ("currenttemperature", "int", "currenttemperature",
 "int"), ("status", "string", "status", "string")), caseSensitive = false,
 transformationContext = "applymapping1")
        // @type: SelectFields
        // @args: [paths = ["sensorid", "currenttemperature", "status"],
 transformation_ctx = "selectfields2"]
        // @return: selectfields2
        // @inputs: [frame = applymapping1]
        val selectfields2 = applymapping1.selectFields(paths = Seq("sensorid",
 "currenttemperature", "status"), transformationContext = "selectfields2")
        // @type: ResolveChoice
        // @args: [choice = "MATCH_CATALOG", database = "tempdb", table_name =
 "my-s3-sink", transformation_ctx = "resolvechoice3"]
```

```
        // @return: resolvechoice3
        // @inputs: [frame = selectfields2]
        val resolvechoice3 = selectfields2.resolveChoice(choiceOption =
 Some(ChoiceOption("MATCH_CATALOG")), database = Some("tempdb"), tableName =
 Some("my-s3-sink"), transformationContext = "resolvechoice3")
        // @type: DataSink
        // @args: [database = "tempdb", table_name = "my-s3-sink",
 transformation_ctx = "datasink4"]
        // @return: datasink4
        // @inputs: [frame = resolvechoice3]
        val datasink4 = glueContext.getCatalogSink(database = "tempdb",
 tableName = "my-s3-sink", redshiftTmpDir = "", transformationContext =
 "datasink4").writeDynamicFrame(resolvechoice3)
        Job.commit()
    }
}
```

Salida:

```
{
    "Name": "my-testing-job"
}
```

Para obtener más información, consulte Creación de trabajos en AWS Glue en la Guía para desarrolladores de AWS Glue.

- Para obtener más información sobre la API, consulte CreateJobla Referencia de AWS CLI comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
const createJob = (name, role, scriptBucketName, scriptKey) => {
  const client = new GlueClient({});
```

```
  const command = new CreateJobCommand({
    Name: name,
    Role: role,
    Command: {
      Name: "glueetl",
      PythonVersion: "3",
      ScriptLocation: `s3://${scriptBucketName}/${scriptKey}`,
    },
    GlueVersion: "3.0",
  });

  return client.send(command);
};
```

- Para obtener más información sobre la API, consulta CreateJobla Referencia AWS SDK for JavaScript de la API.

PHP

SDK para PHP

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
        $role = $iamService->getRole("AWSGlueServiceRole-DocExample");

        $jobName = 'test-job-' . $uniqid;

        $scriptLocation = "s3://$bucketName/run_job.py";
        $job = $glueService->createJob($jobName, $role['Role']['Arn'],
$scriptLocation);

    public function createJob($jobName, $role, $scriptLocation, $pythonVersion =
'3', $glueVersion = '3.0'): Result
    {
        return $this->glueClient->createJob([
```

```
            'Name' => $jobName,
            'Role' => $role,
            'Command' => [
                'Name' => 'glueetl',
                'ScriptLocation' => $scriptLocation,
                'PythonVersion' => $pythonVersion,
            ],
            'GlueVersion' => $glueVersion,
        ]);
    }
```

- Para obtener más información sobre la API, consulta CreateJobla Referencia AWS SDK for PHP de la API.

PowerShell

Herramientas para PowerShell

Ejemplo 1: En este ejemplo se crea un nuevo trabajo en AWS Glue. El valor del nombre del comando es siempre**`glueetl`**. AWS Glue admite la ejecución de scripts de trabajo escritos en Python o Scala. En este ejemplo, el script de trabajo (MyTestGlueJob.py) está escrito en Python. Los parámetros de Python se especifican en la **`$DefArgs`** variable y, a continuación, se pasan al PowerShell comando del **`DefaultArguments`** parámetro, que acepta una tabla hash. Los parámetros de la **`$JobParams`** variable provienen de la CreateJob API, que se documenta en el tema Jobs (https://docs.aws.amazon.com/glue/latest/dg/ aws-glue-api-jobs - job.html) de la referencia de la API de AWS Glue.

```
$Command = New-Object Amazon.Glue.Model.JobCommand
$Command.Name = 'glueetl'
$Command.ScriptLocation = 's3://aws-glue-scripts-000000000000-us-west-2/admin/
MyTestGlueJob.py'
$Command

$Source = "source_test_table"
$Target = "target_test_table"
$Connections = $Source, $Target

$DefArgs = @{
     '--TempDir' = 's3://aws-glue-temporary-000000000000-us-west-2/admin'
     '--job-bookmark-option' = 'job-bookmark-disable'
```

```
     '--job-language' = 'python'
     }
$DefArgs

$ExecutionProp = New-Object Amazon.Glue.Model.ExecutionProperty
$ExecutionProp.MaxConcurrentRuns = 1
$ExecutionProp

$JobParams = @{
    "AllocatedCapacity"    = "5"
    "Command"              = $Command
    "Connections_Connection" = $Connections
    "DefaultArguments"  = $DefArgs
    "Description"       = "This is a test"
    "ExecutionProperty" = $ExecutionProp
    "MaxRetries"        = "1"
    "Name"              = "MyOregonTestGlueJob"
    "Role"              = "Amazon-GlueServiceRoleForSSM"
    "Timeout"           = "20"
     }

New-GlueJob @JobParams
```

- Para obtener más información sobre la API, consulte CreateJobla referencia de AWS Tools for PowerShell cmdlets.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
```

```
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def create_job(self, name, description, role_arn, script_location):
        """
        Creates a job definition for an extract, transform, and load (ETL) job
that can
        be run by AWS Glue.

        :param name: The name of the job definition.
        :param description: The description of the job definition.
        :param role_arn: The ARN of an IAM role that grants AWS Glue the
permissions
                         it requires to run the job.
        :param script_location: The Amazon S3 URL of a Python ETL script that is
run as
                                part of the job. The script defines how the data
is
                                transformed.
        """
        try:
            self.glue_client.create_job(
                Name=name,
                Description=description,
                Role=role_arn,
                Command={
                    "Name": "glueetl",
                    "ScriptLocation": script_location,
                    "PythonVersion": "3",
                },
                GlueVersion="3.0",
            )
        except ClientError as err:
            logger.error(
                "Couldn't create job %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obtener más información sobre la API, consulta CreateJobla AWS Referencia de API de SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Creates a new job with the specified configuration.
  #
  # @param name [String] The name of the job.
  # @param description [String] The description of the job.
  # @param role_arn [String] The ARN of the IAM role to be used by the job.
  # @param script_location [String] The location of the ETL script for the job.
  # @return [void]
  def create_job(name, description, role_arn, script_location)
    @glue_client.create_job(
      name: name,
      description: description,
      role: role_arn,
      command: {
```

```
        name: "glueetl",
        script_location: script_location,
        python_version: "3"
      },
      glue_version: "3.0"
    )
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not create job #{name}: \n#{e.message}")
    raise
  end
```

- Para obtener más información sobre la API, consulta CreateJobla Referencia AWS SDK for Ruby de la API.

Rust

SDK para Rust

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
        let create_job = glue
            .create_job()
            .name(self.job())
            .role(self.iam_role.expose_secret())
            .command(
                JobCommand::builder()
                    .name("glueetl")
                    .python_version("3")
                    .script_location(format!("s3://{}/job.py", self.bucket()))
                    .build(),
            )
            .glue_version("3.0")
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;
```

```
        let job_name = create_job.name().ok_or_else(|| {
            GlueMvpError::Unknown("Did not get job name after creating
job".into())
        })?;
```

- Para obtener más información sobre la API, consulta CreateJobla referencia sobre la API de AWS SDK para Rust.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de este servicio con un AWS SDK. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`DeleteCrawler`** con un AWS SDK o CLI

En los siguientes ejemplos de código, se muestra cómo utilizar `DeleteCrawler`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- Comenzar a ejecutar rastreadores y trabajos

.NET

AWS SDK for .NET

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    /// <summary>
    /// Delete an AWS Glue crawler.
    /// </summary>
    /// <param name="crawlerName">The name of the crawler.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteCrawlerAsync(string crawlerName)
    {
```

```
        var response = await _amazonGlue.DeleteCrawlerAsync(new
DeleteCrawlerRequest { Name = crawlerName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Para obtener más información sobre la API, consulta DeleteCrawlerla Referencia AWS SDK for .NET de la API.

C++

SDK para C++

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::DeleteCrawlerRequest request;
        request.SetName(crawler);

        Aws::Glue::Model::DeleteCrawlerOutcome outcome =
client.DeleteCrawler(request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully deleted the crawler." << std::endl;
        }
        else {
            std::cerr << "Error deleting the crawler. "
                      << outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
```

- Para obtener más información sobre la API, consulta DeleteCrawlerla Referencia AWS SDK for C++ de la API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
const deleteCrawler = (crawlerName) => {
  const client = new GlueClient({});

  const command = new DeleteCrawlerCommand({
    Name: crawlerName,
  });

  return client.send(command);
};
```

- Para obtener más información sobre la API, consulta DeleteCrawlerla Referencia AWS SDK for JavaScript de la API.

PHP

SDK para PHP

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
        echo "Delete the crawler.\n";
        $glueClient->deleteCrawler([
            'Name' => $crawlerName,
        ]);

    public function deleteCrawler($crawlerName)
    {
        return $this->glueClient->deleteCrawler([
            'Name' => $crawlerName,
        ]);
    }
```

- Para obtener más información sobre la API, consulta DeleteCrawlerla Referencia AWS SDK for PHP de la API.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def delete_crawler(self, name):
        """
        Deletes a crawler.

        :param name: The name of the crawler to delete.
```

```
        """
        try:
            self.glue_client.delete_crawler(Name=name)
        except ClientError as err:
            logger.error(
                "Couldn't delete crawler %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obtener más información sobre la API, consulta DeleteCrawlerla AWS Referencia de API de SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Deletes a crawler with the specified name.
```

```
  #
  # @param name [String] The name of the crawler to delete.
  # @return [void]
  def delete_crawler(name)
    @glue_client.delete_crawler(name: name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete crawler #{name}: \n#{e.message}")
    raise
  end
```

- Para obtener más información sobre la API, consulta DeleteCrawlerla Referencia AWS SDK for Ruby de la API.

Rust

SDK para Rust

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
        glue.delete_crawler()
            .name(self.crawler())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;
```

- Para obtener más información sobre la API, consulta DeleteCrawlerla referencia sobre la API de AWS SDK para Rust.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de este servicio con un AWS SDK. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`DeleteDatabase`** con un AWS SDK o CLI

En los siguientes ejemplos de código, se muestra cómo utilizar `DeleteDatabase`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- Comenzar a ejecutar rastreadores y trabajos

.NET

AWS SDK for .NET

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    /// <summary>
    /// Delete the AWS Glue database.
    /// </summary>
    /// <param name="dbName">The name of the database.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteDatabaseAsync(string dbName)
    {
        var response = await _amazonGlue.DeleteDatabaseAsync(new
DeleteDatabaseRequest { Name = dbName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Para obtener más información sobre la API, consulta DeleteDatabasela Referencia AWS SDK for .NET de la API.

C++

SDK para C++

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::DeleteDatabaseRequest request;
        request.SetName(database);

        Aws::Glue::Model::DeleteDatabaseOutcome outcome = client.DeleteDatabase(
                request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully deleted the database." << std::endl;
        }
        else {
            std::cerr << "Error deleting database. " <<
outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
```

- Para obtener más información sobre la API, consulta DeleteDatabasela Referencia AWS SDK for C++ de la API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
const deleteDatabase = (databaseName) => {
  const client = new GlueClient({});

  const command = new DeleteDatabaseCommand({
    Name: databaseName,
  });

  return client.send(command);
};
```

- Para obtener más información sobre la API, consulta DeleteDatabasela Referencia AWS SDK for JavaScript de la API.

PHP

SDK para PHP

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
        echo "Delete the databases.\n";
        $glueClient->deleteDatabase([
            'Name' => $databaseName,
        ]);

    public function deleteDatabase($databaseName)
```

```
    {
        return $this->glueClient->deleteDatabase([
            'Name' => $databaseName,
        ]);
    }
```

- Para obtener más información sobre la API, consulta DeleteDatabasela Referencia AWS SDK for PHP de la API.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def delete_database(self, name):
        """
        Deletes a metadata database from your Data Catalog.

        :param name: The name of the database to delete.
        """
        try:
            self.glue_client.delete_database(Name=name)
        except ClientError as err:
            logger.error(
                "Couldn't delete database %s. Here's why: %s: %s",
```

```
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obtener más información sobre la API, consulta DeleteDatabasela AWS Referencia de API de SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Removes a specified database from a Data Catalog.
  #
  # @param database_name [String] The name of the database to delete.
  # @return [void]
  def delete_database(database_name)
    @glue_client.delete_database(name: database_name)
  rescue Aws::Glue::Errors::ServiceError => e
```

```
    @logger.error("Glue could not delete database: \n#{e.message}")
  end
```

- Para obtener más información sobre la API, consulta DeleteDatabasela Referencia AWS SDK for Ruby de la API.

Rust

SDK para Rust

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
        glue.delete_database()
            .name(self.database())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;
```

- Para obtener más información sobre la API, consulta DeleteDatabasela referencia sobre la API de AWS SDK para Rust.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de este servicio con un AWS SDK. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`DeleteJob`** con un AWS SDK o CLI

En los siguientes ejemplos de código, se muestra cómo utilizar `DeleteJob`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- Comenzar a ejecutar rastreadores y trabajos

.NET

AWS SDK for .NET

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    /// <summary>
    /// Delete an AWS Glue job.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteJobAsync(string jobName)
    {
        var response = await _amazonGlue.DeleteJobAsync(new DeleteJobRequest
{ JobName = jobName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Para obtener más información sobre la API, consulta DeleteJobla Referencia AWS SDK for .NET de la API.

C++

SDK para C++

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
```

```
        // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::DeleteJobRequest request;
        request.SetJobName(job);

        Aws::Glue::Model::DeleteJobOutcome outcome = client.DeleteJob(request);


        if (outcome.IsSuccess()) {
            std::cout << "Successfully deleted the job." << std::endl;
        }
        else {
            std::cerr << "Error deleting the job. " <<
 outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
```

- Para obtener más información sobre la API, consulta DeleteJobla Referencia AWS SDK for C++ de la API.

CLI

AWS CLI

Para eliminar un trabajo

En el siguiente ejemplo de `delete-job`, se elimina un trabajo que ya no necesite.

```
aws glue delete-job \
    --job-name my-testing-job
```

Salida:

```
{
    "JobName": "my-testing-job"
}
```

Para obtener más información, consulte Cómo trabajar con trabajos en la consola de AWS Glue en la Guía para desarrolladores de AWS Glue.

- Para obtener más información sobre la API, consulte DeleteJobla Referencia de AWS CLI comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
const deleteJob = (jobName) => {
  const client = new GlueClient({});

  const command = new DeleteJobCommand({
    JobName: jobName,
  });

  return client.send(command);
};
```

- Para obtener más información sobre la API, consulta DeleteJobla Referencia AWS SDK for JavaScript de la API.

PHP

SDK para PHP

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
        echo "Delete the job.\n";
        $glueClient->deleteJob([
            'JobName' => $job['Name'],
        ]);

    public function deleteJob($jobName)
    {
        return $this->glueClient->deleteJob([
            'JobName' => $jobName,
        ]);
    }
```

- Para obtener más información sobre la API, consulta DeleteJobla Referencia AWS SDK for PHP de la API.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def delete_job(self, job_name):
        """
        Deletes a job definition. This also deletes data about all runs that are
        associated with this job definition.
```

```
        :param job_name: The name of the job definition to delete.
        """
        try:
            self.glue_client.delete_job(JobName=job_name)
        except ClientError as err:
            logger.error(
                "Couldn't delete job %s. Here's why: %s: %s",
                job_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obtener más información sobre la API, consulta DeleteJobla AWS Referencia de API de SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end
```

```
  # Deletes a job with the specified name.
  #
  # @param job_name [String] The name of the job to delete.
  # @return [void]
  def delete_job(job_name)
    @glue_client.delete_job(job_name: job_name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete job: \n#{e.message}")
  end
```

- Para obtener más información sobre la API, consulta DeleteJobla Referencia AWS SDK for Ruby de la API.

Rust

SDK para Rust

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
        glue.delete_job()
            .job_name(self.job())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;
```

- Para obtener más información sobre la API, consulta DeleteJobla referencia sobre la API de AWS SDK para Rust.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de este servicio con un AWS SDK. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`DeleteTable`** con un AWS SDK o CLI

En los siguientes ejemplos de código, se muestra cómo utilizar `DeleteTable`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- Comenzar a ejecutar rastreadores y trabajos

.NET

AWS SDK for .NET

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    /// <summary>
    /// Delete a table from an AWS Glue database.
    /// </summary>
    /// <param name="tableName">The table to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteTableAsync(string dbName, string tableName)
    {
        var response = await _amazonGlue.DeleteTableAsync(new DeleteTableRequest
{ Name = tableName, DatabaseName = dbName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- Para obtener más información sobre la API, consulta DeleteTablela Referencia AWS SDK for .NET de la API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
const deleteTable = (databaseName, tableName) => {
  const client = new GlueClient({});

  const command = new DeleteTableCommand({
    DatabaseName: databaseName,
    Name: tableName,
  });

  return client.send(command);
};
```

- Para obtener más información sobre la API, consulta DeleteTablela Referencia AWS SDK for JavaScript de la API.

PHP

SDK para PHP

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
        echo "Delete the tables.\n";
        foreach ($tables['TableList'] as $table) {
            $glueService->deleteTable($table['Name'], $databaseName);
        }
```

```
    public function deleteTable($tableName, $databaseName)
    {
        return $this->glueClient->deleteTable([
            'DatabaseName' => $databaseName,
            'Name' => $tableName,
        ]);
    }
```

- Para obtener más información sobre la API, consulta DeleteTablela Referencia AWS SDK for PHP de la API.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def delete_table(self, db_name, table_name):
        """
        Deletes a table from a metadata database.

        :param db_name: The name of the database that contains the table.
        :param table_name: The name of the table to delete.
        """
        try:
            self.glue_client.delete_table(DatabaseName=db_name, Name=table_name)
```

```
        except ClientError as err:
            logger.error(
                "Couldn't delete table %s. Here's why: %s: %s",
                table_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obtener más información sobre la API, consulta DeleteTablela AWS Referencia de API de SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Deletes a table with the specified name.
  #
  # @param database_name [String] The name of the catalog database in which the
 table resides.
```

```
  # @param table_name [String] The name of the table to be deleted.
  # @return [void]
  def delete_table(database_name, table_name)
    @glue_client.delete_table(database_name: database_name, name: table_name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete job: \n#{e.message}")
  end
```

- Para obtener más información sobre la API, consulta DeleteTablela Referencia AWS SDK for Ruby de la API.

Rust

SDK para Rust

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
        for t in &self.tables {
            glue.delete_table()
                .name(t.name())
                .database_name(self.database())
                .send()
                .await
                .map_err(GlueMvpError::from_glue_sdk)?;
        }
```

- Para obtener más información sobre la API, consulta DeleteTablela referencia sobre la API de AWS SDK para Rust.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de este servicio con un AWS SDK. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`GetCrawler`** con un AWS SDK o CLI

En los siguientes ejemplos de código, se muestra cómo utilizar `GetCrawler`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- Comenzar a ejecutar rastreadores y trabajos

.NET

AWS SDK for .NET

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    /// <summary>
    /// Get information about an AWS Glue crawler.
    /// </summary>
    /// <param name="crawlerName">The name of the crawler.</param>
    /// <returns>A Crawler object describing the crawler.</returns>
    public async Task<Crawler?> GetCrawlerAsync(string crawlerName)
    {
        var crawlerRequest = new GetCrawlerRequest
        {
            Name = crawlerName,
        };

        var response = await _amazonGlue.GetCrawlerAsync(crawlerRequest);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            var databaseName = response.Crawler.DatabaseName;
            Console.WriteLine($"{crawlerName} has the database {databaseName}");
            return response.Crawler;
        }

        Console.WriteLine($"No information regarding {crawlerName} could be
found.");
```

```
        return null;
    }
```

- Para obtener más información sobre la API, consulta GetCrawlerla Referencia AWS SDK for .NET de la API.

C++

SDK para C++

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
       Aws::Client::ClientConfiguration clientConfig;
       // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
       // clientConfig.region = "us-east-1";

   Aws::Glue::GlueClient client(clientConfig);

       Aws::Glue::Model::GetCrawlerRequest request;
       request.SetName(CRAWLER_NAME);

       Aws::Glue::Model::GetCrawlerOutcome outcome = client.GetCrawler(request);

       if (outcome.IsSuccess()) {
           Aws::Glue::Model::CrawlerState crawlerState =
outcome.GetResult().GetCrawler().GetState();
           std::cout << "Retrieved crawler with state " <<

Aws::Glue::Model::CrawlerStateMapper::GetNameForCrawlerState(
                           crawlerState)
                  << "." << std::endl;
       }
       else {
           std::cerr << "Error retrieving a crawler.  "
```

```
                    << outcome.GetError().GetMessage() << std::endl;
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                         clientConfig);
            return false;
        }
```

- Para obtener más información sobre la API, consulta GetCrawlerla Referencia AWS SDK for C++ de la API.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.GetCrawlerRequest;
import software.amazon.awssdk.services.glue.model.GetCrawlerResponse;
import software.amazon.awssdk.services.glue.model.GlueException;
import java.time.Instant;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;
import java.util.Locale;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class GetCrawler {
```

```
   public static void main(String[] args) {
       final String usage = """

               Usage:
                   <crawlerName>

               Where:
                   crawlerName - The name of the crawler.\s
               """;

       if (args.length != 1) {
           System.out.println(usage);
           System.exit(1);
       }

       String crawlerName = args[0];
       Region region = Region.US_EAST_1;
       GlueClient glueClient = GlueClient.builder()
               .region(region)
               .build();

       getSpecificCrawler(glueClient, crawlerName);
       glueClient.close();
   }

   public static void getSpecificCrawler(GlueClient glueClient, String
crawlerName) {
       try {
           GetCrawlerRequest crawlerRequest = GetCrawlerRequest.builder()
                   .name(crawlerName)
                   .build();

           GetCrawlerResponse response = glueClient.getCrawler(crawlerRequest);
           Instant createDate = response.crawler().creationTime();

           // Convert the Instant to readable date
           DateTimeFormatter formatter =
DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
                   .withLocale(Locale.US)
                   .withZone(ZoneId.systemDefault());

           formatter.format(createDate);
           System.out.println("The create date of the Crawler is " +
createDate);
```

```
        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obtener más información sobre la API, consulta GetCrawlerla Referencia AWS SDK for Java 2.x de la API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
const getCrawler = (name) => {
  const client = new GlueClient({});

  const command = new GetCrawlerCommand({
    Name: name,
  });

  return client.send(command);
};
```

- Para obtener más información sobre la API, consulta GetCrawlerla Referencia AWS SDK for JavaScript de la API.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun getSpecificCrawler(crawlerName: String?) {
    val request =
        GetCrawlerRequest {
            name = crawlerName
        }
    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getCrawler(request)
        val role = response.crawler?.role
        println("The role associated with this crawler is $role")
    }
}
```

- Para obtener más información sobre la API, consulta GetCrawlerla referencia sobre el AWS SDK para la API de Kotlin.

PHP

SDK para PHP

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
        echo "Waiting for crawler";
        do {
            $crawler = $glueService->getCrawler($crawlerName);
```

```
            echo ".";
            sleep(10);
        } while ($crawler['Crawler']['State'] != "READY");
        echo "\n";

    public function getCrawler($crawlerName)
    {
        return $this->customWaiter(function () use ($crawlerName) {
            return $this->glueClient->getCrawler([
                'Name' => $crawlerName,
            ]);
        });
    }
```

- Para obtener más información sobre la API, consulta GetCrawlerla Referencia AWS SDK for PHP de la API.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def get_crawler(self, name):
        """
        Gets information about a crawler.
```

```
        :param name: The name of the crawler to look up.
        :return: Data about the crawler.
        """
        crawler = None
        try:
            response = self.glue_client.get_crawler(Name=name)
            crawler = response["Crawler"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityNotFoundException":
                logger.info("Crawler %s doesn't exist.", name)
            else:
                logger.error(
                    "Couldn't get crawler %s. Here's why: %s: %s",
                    name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return crawler
```

- Para obtener más información sobre la API, consulta GetCrawlerla AWS Referencia de API de SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
```

```
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Retrieves information about a specific crawler.
  #
  # @param name [String] The name of the crawler to retrieve information about.
  # @return [Aws::Glue::Types::Crawler, nil] The crawler object if found, or nil
 if not found.
  def get_crawler(name)
    @glue_client.get_crawler(name: name)
  rescue Aws::Glue::Errors::EntityNotFoundException
    @logger.info("Crawler #{name} doesn't exist.")
    false
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get crawler #{name}: \n#{e.message}")
    raise
  end
```

- Para obtener más información sobre la API, consulta GetCrawlerla Referencia AWS SDK for Ruby de la API.

Rust

SDK para Rust

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
            let tmp_crawler = glue
                .get_crawler()
                .name(self.crawler())
                .send()
```

```
                .await
                .map_err(GlueMvpError::from_glue_sdk)?;
```

- Para obtener más información sobre la API, consulta GetCrawlerla referencia sobre la API de AWS SDK para Rust.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de este servicio con un AWS SDK. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`GetDatabase`** con un AWS SDK o CLI

En los siguientes ejemplos de código, se muestra cómo utilizar `GetDatabase`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- Comenzar a ejecutar rastreadores y trabajos

.NET

AWS SDK for .NET

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    /// <summary>
    /// Get information about an AWS Glue database.
    /// </summary>
    /// <param name="dbName">The name of the database.</param>
    /// <returns>A Database object containing information about the database.</
returns>
    public async Task<Database> GetDatabaseAsync(string dbName)
    {
        var databasesRequest = new GetDatabaseRequest
```

```
        {
            Name = dbName,
        };

        var response = await _amazonGlue.GetDatabaseAsync(databasesRequest);
        return response.Database;
    }
```

- Para obtener más información sobre la API, consulta GetDatabasela Referencia AWS SDK for .NET de la API.

C++

SDK para C++

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
       Aws::Client::ClientConfiguration clientConfig;
       // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
       // clientConfig.region = "us-east-1";

   Aws::Glue::GlueClient client(clientConfig);

       Aws::Glue::Model::GetDatabaseRequest request;
       request.SetName(CRAWLER_DATABASE_NAME);

       Aws::Glue::Model::GetDatabaseOutcome outcome =
client.GetDatabase(request);

       if (outcome.IsSuccess()) {
           const Aws::Glue::Model::Database &database =
outcome.GetResult().GetDatabase();

           std::cout << "Successfully retrieve the database\n" <<
```

```
                    database.Jsonize().View().WriteReadable() << "'." <<
std::endl;
        }
        else {
            std::cerr << "Error getting the database.  "
                      << outcome.GetError().GetMessage() << std::endl;
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                         clientConfig);
            return false;
        }
```

- Para obtener más información sobre la API, consulta GetDatabasela Referencia AWS SDK for C++ de la API.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.GetDatabaseRequest;
import software.amazon.awssdk.services.glue.model.GetDatabaseResponse;
import software.amazon.awssdk.services.glue.model.GlueException;
import java.time.Instant;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;
import java.util.Locale;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
```

```
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class GetDatabase {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <databaseName>

                Where:
                    databaseName - The name of the database.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String databaseName = args[0];
        Region region = Region.US_EAST_1;
        GlueClient glueClient = GlueClient.builder()
                .region(region)
                .build();

        getSpecificDatabase(glueClient, databaseName);
        glueClient.close();
    }

    public static void getSpecificDatabase(GlueClient glueClient, String
 databaseName) {
        try {
            GetDatabaseRequest databasesRequest = GetDatabaseRequest.builder()
                    .name(databaseName)
                    .build();

            GetDatabaseResponse response =
 glueClient.getDatabase(databasesRequest);
            Instant createDate = response.database().createTime();

            // Convert the Instant to readable date.
            DateTimeFormatter formatter =
 DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
```

```
                    .withLocale(Locale.US)
                    .withZone(ZoneId.systemDefault());

            formatter.format(createDate);
            System.out.println("The create date of the database is " +
 createDate);

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obtener más información sobre la API, consulta GetDatabasela Referencia AWS SDK for Java 2.x de la API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
const getDatabase = (name) => {
  const client = new GlueClient({});

  const command = new GetDatabaseCommand({
    Name: name,
  });

  return client.send(command);
};
```

- Para obtener más información sobre la API, consulta GetDatabasela Referencia AWS SDK for JavaScript de la API.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun getSpecificDatabase(databaseName: String?) {
    val request =
        GetDatabaseRequest {
            name = databaseName
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getDatabase(request)
        val dbDesc = response.database?.description
        println("The database description is $dbDesc")
    }
}
```

- Para obtener más información sobre la API, consulta GetDatabasela referencia sobre el AWS SDK para la API de Kotlin.

PHP

SDK para PHP

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
        $databaseName = "doc-example-database-$uniqid";
```

```
        $database = $glueService->getDatabase($databaseName);
        echo "Found a database named " . $database['Database']['Name'] . "\n";

    public function getDatabase(string $databaseName): Result
    {
        return $this->customWaiter(function () use ($databaseName) {
            return $this->glueClient->getDatabase([
                'Name' => $databaseName,
            ]);
        });
    }
```

- Para obtener más información sobre la API, consulta GetDatabasela Referencia AWS SDK for PHP de la API.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def get_database(self, name):
        """
        Gets information about a database in your Data Catalog.

        :param name: The name of the database to look up.
```

```
        :return: Information about the database.
        """
        try:
            response = self.glue_client.get_database(Name=name)
        except ClientError as err:
            logger.error(
                "Couldn't get database %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["Database"]
```

- Para obtener más información sobre la API, consulta GetDatabasela AWS Referencia de API de SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
```

```
  end

  # Retrieves information about a specific database.
  #
  # @param name [String] The name of the database to retrieve information about.
  # @return [Aws::Glue::Types::Database, nil] The database object if found, or
 nil if not found.
  def get_database(name)
    response = @glue_client.get_database(name: name)
    response.database
rescue Aws::Glue::Errors::GlueException => e
  @logger.error("Glue could not get database #{name}: \n#{e.message}")
  raise
  end
```

- Para obtener más información sobre la API, consulta GetDatabasela Referencia AWS SDK for Ruby de la API.

Rust

SDK para Rust

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
        let database = glue
            .get_database()
            .name(self.database())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?
            .to_owned();
        let database = database
            .database()
            .ok_or_else(|| GlueMvpError::Unknown("Could not find
 database".into()))?;
```

- Para obtener más información sobre la API, consulta GetDatabasela referencia sobre la API de AWS SDK para Rust.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de este servicio con un AWS SDK. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`GetDatabases`** con un AWS SDK o CLI

En los siguientes ejemplos de código, se muestra cómo utilizar `GetDatabases`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- Comenzar a ejecutar rastreadores y trabajos

CLI

AWS CLI

Para enumerar las definiciones de algunas o todas las bases de datos del catálogo de datos de AWS Glue

El siguiente ejemplo de `get-databases` devuelve información sobre las bases de datos del Catálogo de datos.

```
aws glue get-databases
```

Salida:

```
{
    "DatabaseList": [
        {
            "Name": "default",
            "Description": "Default Hive database",
            "LocationUri": "file:/spark-warehouse",
            "CreateTime": 1602084052.0,
            "CreateTableDefaultPermissions": [
                {
                    "Principal": {
                        "DataLakePrincipalIdentifier": "IAM_ALLOWED_PRINCIPALS"
```

```
                    },
                    "Permissions": [
                        "ALL"
                    ]
                }
            ],
            "CatalogId": "111122223333"
        },
        {
            "Name": "flights-db",
            "CreateTime": 1587072847.0,
            "CreateTableDefaultPermissions": [
                {
                    "Principal": {
                        "DataLakePrincipalIdentifier": "IAM_ALLOWED_PRINCIPALS"
                    },
                    "Permissions": [
                        "ALL"
                    ]
                }
            ],
            "CatalogId": "111122223333"
        },
        {
            "Name": "legislators",
            "CreateTime": 1601415625.0,
            "CreateTableDefaultPermissions": [
                {
                    "Principal": {
                        "DataLakePrincipalIdentifier": "IAM_ALLOWED_PRINCIPALS"
                    },
                    "Permissions": [
                        "ALL"
                    ]
                }
            ],
            "CatalogId": "111122223333"
        },
        {
            "Name": "tempdb",
            "CreateTime": 1601498566.0,
            "CreateTableDefaultPermissions": [
                {
                    "Principal": {
```

```
                        "DataLakePrincipalIdentifier": "IAM_ALLOWED_PRINCIPALS"
                    },
                    "Permissions": [
                        "ALL"
                    ]
                }
            ],
            "CatalogId": "111122223333"
        }
    ]
}
```

Para obtener más información, consulte Definición de una base de datos en su Catálogo de datos en la Guía para desarrolladores de AWS Glue.

- Para obtener más información sobre la API, consulte GetDatabasesla Referencia de AWS CLI comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
const getDatabases = () => {
  const client = new GlueClient({});

  const command = new GetDatabasesCommand({});

  return client.send(command);
};
```

- Para obtener más información sobre la API, consulta GetDatabasesla Referencia AWS SDK for JavaScript de la API.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de este servicio con un AWS SDK. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`GetJob`** con un AWS SDK o CLI

En los siguientes ejemplos de código, se muestra cómo utilizar `GetJob`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- Comenzar a ejecutar rastreadores y trabajos

CLI

AWS CLI

Para recuperar información sobre un trabajo

El siguiente ejemplo de `get-job` recupera información sobre un trabajo.

```
aws glue get-job \
    --job-name my-testing-job
```

Salida:

```
{
    "Job": {
        "Name": "my-testing-job",
        "Role": "Glue_DefaultRole",
        "CreatedOn": 1602805698.167,
        "LastModifiedOn": 1602805698.167,
        "ExecutionProperty": {
            "MaxConcurrentRuns": 1
        },
        "Command": {
            "Name": "gluestreaming",
            "ScriptLocation": "s3://janetst-bucket-01/Scripts/test_script.scala",
            "PythonVersion": "2"
        },
        "DefaultArguments": {
```

```
            "--class": "GlueApp",
            "--job-language": "scala"
        },
        "MaxRetries": 0,
        "AllocatedCapacity": 10,
        "MaxCapacity": 10.0,
        "GlueVersion": "1.0"
    }
}
```

Para obtener más información, consulte Trabajos en la Guía para desarrolladores de AWS Glue.

- Para obtener más información sobre la API, consulte GetJobla Referencia de AWS CLI comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
const getJob = (jobName) => {
  const client = new GlueClient({});

  const command = new GetJobCommand({
    JobName: jobName,
  });

  return client.send(command);
};
```

- Para obtener más información sobre la API, consulta GetJobla Referencia AWS SDK for JavaScript de la API.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de este servicio con un AWS SDK. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **GetJobRun** con un AWS SDK o CLI

En los siguientes ejemplos de código, se muestra cómo utilizar `GetJobRun`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- Comenzar a ejecutar rastreadores y trabajos

.NET

AWS SDK for .NET

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    /// <summary>
    /// Get information about a specific AWS Glue job run.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <param name="jobRunId">The Id of the job run.</param>
    /// <returns>A JobRun object with information about the job run.</returns>
    public async Task<JobRun> GetJobRunAsync(string jobName, string jobRunId)
    {
        var response = await _amazonGlue.GetJobRunAsync(new GetJobRunRequest
{ JobName = jobName, RunId = jobRunId });
        return response.JobRun;
    }
```

- Para obtener más información sobre la API, consulta GetJobRunla Referencia AWS SDK for .NET de la API.

C++

SDK para C++

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::GetJobRunRequest jobRunRequest;
        jobRunRequest.SetJobName(jobName);
        jobRunRequest.SetRunId(jobRunID);

        Aws::Glue::Model::GetJobRunOutcome jobRunOutcome = client.GetJobRun(
                jobRunRequest);

        if (jobRunOutcome.IsSuccess()) {
            std::cout << "Displaying the job run JSON description." << std::endl;
            std::cout
                    <<
jobRunOutcome.GetResult().GetJobRun().Jsonize().View().WriteReadable()
                    << std::endl;
        }
        else {
            std::cerr << "Error get a job run. "
                      << jobRunOutcome.GetError().GetMessage()
                      << std::endl;
        }
```

- Para obtener más información sobre la API, consulta GetJobRunla Referencia AWS SDK for C++ de la API.

CLI

AWS CLI

Para obtener información sobre una ejecución de trabajo

El siguiente ejemplo de `get-job-run` recupera información sobre una ejecución de trabajo.

```
aws glue get-job-run \
    --job-name "Combine legistators data" \
    --run-id jr_012e176506505074d94d761755e5c62538ee1aad6f17d39f527e9140cf0c9a5e
```

Salida:

```
{
    "JobRun": {
        "Id":
 "jr_012e176506505074d94d761755e5c62538ee1aad6f17d39f527e9140cf0c9a5e",
        "Attempt": 0,
        "JobName": "Combine legistators data",
        "StartedOn": 1602873931.255,
        "LastModifiedOn": 1602874075.985,
        "CompletedOn": 1602874075.985,
        "JobRunState": "SUCCEEDED",
        "Arguments": {
            "--enable-continuous-cloudwatch-log": "true",
            "--enable-metrics": "",
            "--enable-spark-ui": "true",
            "--job-bookmark-option": "job-bookmark-enable",
            "--spark-event-logs-path": "s3://aws-glue-assets-111122223333-us-
east-1/sparkHistoryLogs/"
        },
        "PredecessorRuns": [],
        "AllocatedCapacity": 10,
        "ExecutionTime": 117,
        "Timeout": 2880,
        "MaxCapacity": 10.0,
        "WorkerType": "G.1X",
        "NumberOfWorkers": 10,
        "LogGroupName": "/aws-glue/jobs",
        "GlueVersion": "2.0"
    }
}
```

Para obtener más información, consulte Ejecuciones de trabajo en la Guía para desarrolladores de AWS Glue.

- Para obtener más información sobre la API, consulta GetJobRunla Referencia de AWS CLI comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
const getJobRun = (jobName, jobRunId) => {
  const client = new GlueClient({});
  const command = new GetJobRunCommand({
    JobName: jobName,
    RunId: jobRunId,
  });

  return client.send(command);
};
```

- Para obtener más información sobre la API, consulta GetJobRunla Referencia AWS SDK for JavaScript de la API.

PHP

SDK para PHP

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
        $jobName = 'test-job-' . $uniqid;

        $outputBucketUrl = "s3://$bucketName";
        $runId = $glueService->startJobRun($jobName, $databaseName, $tables,
$outputBucketUrl)['JobRunId'];

        echo "waiting for job";
        do {
            $jobRun = $glueService->getJobRun($jobName, $runId);
            echo ".";
            sleep(10);
        } while (!array_intersect([$jobRun['JobRun']['JobRunState']],
['SUCCEEDED', 'STOPPED', 'FAILED', 'TIMEOUT']));
        echo "\n";

    public function getJobRun($jobName, $runId, $predecessorsIncluded = false):
Result
    {
        return $this->glueClient->getJobRun([
            'JobName' => $jobName,
            'RunId' => $runId,
            'PredecessorsIncluded' => $predecessorsIncluded,
        ]);
    }
```

- Para obtener más información sobre la API, consulta GetJobRunla Referencia AWS SDK for PHP de la API.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""
```

```
    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def get_job_run(self, name, run_id):
        """
        Gets information about a single job run.

        :param name: The name of the job definition for the run.
        :param run_id: The ID of the run.
        :return: Information about the run.
        """
        try:
            response = self.glue_client.get_job_run(JobName=name, RunId=run_id)
        except ClientError as err:
            logger.error(
                "Couldn't get job run %s/%s. Here's why: %s: %s",
                name,
                run_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["JobRun"]
```

- Para obtener más información sobre la API, consulta GetJobRunla AWS Referencia de API de SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Retrieves data for a specific job run.
  #
  # @param job_name [String] The name of the job run to retrieve data for.
  # @return [Glue::Types::GetJobRunResponse]
  def get_job_run(job_name, run_id)
    @glue_client.get_job_run(job_name: job_name, run_id: run_id)
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get job runs: \n#{e.message}")
  end
```

- Para obtener más información sobre la API, consulta GetJobRunla Referencia AWS SDK for Ruby de la API.

Rust

SDK para Rust

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
        let get_job_run = || async {
            Ok::<JobRun, GlueMvpError>(
                glue.get_job_run()
                    .job_name(self.job())
                    .run_id(job_run_id.to_string())
                    .send()
                    .await
                    .map_err(GlueMvpError::from_glue_sdk)?
                    .job_run()
                    .ok_or_else(|| GlueMvpError::Unknown("Failed to get
job_run".into()))?
                    .to_owned(),
            )
        };

        let mut job_run = get_job_run().await?;
        let mut state =
job_run.job_run_state().unwrap_or(&unknown_state).to_owned();

        while matches!(
            state,
            JobRunState::Starting | JobRunState::Stopping | JobRunState::Running
        ) {
            info!(?state, "Waiting for job to finish");
            tokio::time::sleep(self.wait_delay).await;

            job_run = get_job_run().await?;
            state = job_run.job_run_state().unwrap_or(&unknown_state).to_owned();
        }
```

- Para obtener más información sobre la API, consulta GetJobRunla referencia sobre la API de AWS SDK para Rust.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de este servicio con un AWS SDK. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`GetJobRuns`** con un AWS SDK o CLI

En los siguientes ejemplos de código, se muestra cómo utilizar `GetJobRuns`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- Comenzar a ejecutar rastreadores y trabajos

.NET

AWS SDK for .NET

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
/// <summary>
/// Get information about all AWS Glue runs of a specific job.
/// </summary>
/// <param name="jobName">The name of the job.</param>
/// <returns>A list of JobRun objects.</returns>
public async Task<List<JobRun>> GetJobRunsAsync(string jobName)
{
    var jobRuns = new List<JobRun>();

    var request = new GetJobRunsRequest
    {
        JobName = jobName,
    };
```

```
        // No need to loop to get all the log groups--the SDK does it for us
behind the scenes
        var paginatorForJobRuns =
            _amazonGlue.Paginators.GetJobRuns(request);

        await foreach (var response in paginatorForJobRuns.Responses)
        {
            response.JobRuns.ForEach(jobRun =>
            {
                jobRuns.Add(jobRun);
            });
        }

        return jobRuns;
    }
```

- Para obtener más información sobre la API, consulta GetJobRunsla Referencia AWS SDK for .NET de la API.

C++

SDK para C++

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::GetJobRunsRequest getJobRunsRequest;
        getJobRunsRequest.SetJobName(jobName);
```

```
        Aws::String nextToken; // Used for pagination.
        std::vector<Aws::Glue::Model::JobRun> allJobRuns;
        do {
            if (!nextToken.empty()) {
                getJobRunsRequest.SetNextToken(nextToken);
            }
            Aws::Glue::Model::GetJobRunsOutcome jobRunsOutcome =
client.GetJobRuns(
                    getJobRunsRequest);

            if (jobRunsOutcome.IsSuccess()) {
                const std::vector<Aws::Glue::Model::JobRun> &jobRuns =
jobRunsOutcome.GetResult().GetJobRuns();
                allJobRuns.insert(allJobRuns.end(), jobRuns.begin(),
jobRuns.end());

                nextToken = jobRunsOutcome.GetResult().GetNextToken();
            }
            else {
                std::cerr << "Error getting job runs. "
                          << jobRunsOutcome.GetError().GetMessage()
                          << std::endl;
                break;
            }
        } while (!nextToken.empty());
```

- Para obtener más información sobre la API, consulta GetJobRunsla Referencia AWS SDK for C++ de la API.

CLI

AWS CLI

Para obtener información sobre todas las ejecuciones de trabajo para un trabajo

El siguiente ejemplo de `get-job-runs` recupera información acerca de las ejecuciones de flujo de trabajo para una tarea.

```
aws glue get-job-runs \
    --job-name "my-testing-job"
```

Salida:

```
{
    "JobRuns": [
        {
            "Id":
 "jr_012e176506505074d94d761755e5c62538ee1aad6f17d39f527e9140cf0c9a5e",
            "Attempt": 0,
            "JobName": "my-testing-job",
            "StartedOn": 1602873931.255,
            "LastModifiedOn": 1602874075.985,
            "CompletedOn": 1602874075.985,
            "JobRunState": "SUCCEEDED",
            "Arguments": {
                "--enable-continuous-cloudwatch-log": "true",
                "--enable-metrics": "",
                "--enable-spark-ui": "true",
                "--job-bookmark-option": "job-bookmark-enable",
                "--spark-event-logs-path": "s3://aws-glue-assets-111122223333-us-
east-1/sparkHistoryLogs/"
            },
            "PredecessorRuns": [],
            "AllocatedCapacity": 10,
            "ExecutionTime": 117,
            "Timeout": 2880,
            "MaxCapacity": 10.0,
            "WorkerType": "G.1X",
            "NumberOfWorkers": 10,
            "LogGroupName": "/aws-glue/jobs",
            "GlueVersion": "2.0"
        },
        {
            "Id":
 "jr_03cc19ddab11c4e244d3f735567de74ff93b0b3ef468a713ffe73e53d1aec08f_attempt_2",
            "Attempt": 2,
            "PreviousRunId":
 "jr_03cc19ddab11c4e244d3f735567de74ff93b0b3ef468a713ffe73e53d1aec08f_attempt_1",
            "JobName": "my-testing-job",
            "StartedOn": 1602811168.496,
            "LastModifiedOn": 1602811282.39,
            "CompletedOn": 1602811282.39,
            "JobRunState": "FAILED",
            "ErrorMessage": "An error occurred while calling
 o122.pyWriteDynamicFrame.
```

```
                Access Denied (Service: Amazon S3; Status Code: 403; Error Code:
 AccessDenied;
                Request ID: 021AAB703DB20A2D;
                S3 Extended Request ID: teZk24Y09TkXzBvMPG502L5VJBhe9DJuWA9/
TXtuGOqfByajkfL/Tlqt5JBGdEGpigAqzdMDM/U=)",
            "PredecessorRuns": [],
            "AllocatedCapacity": 10,
            "ExecutionTime": 110,
            "Timeout": 2880,
            "MaxCapacity": 10.0,
            "WorkerType": "G.1X",
            "NumberOfWorkers": 10,
            "LogGroupName": "/aws-glue/jobs",
            "GlueVersion": "2.0"
        },
        {
            "Id":
 "jr_03cc19ddab11c4e244d3f735567de74ff93b0b3ef468a713ffe73e53d1aec08f_attempt_1",
            "Attempt": 1,
            "PreviousRunId":
 "jr_03cc19ddab11c4e244d3f735567de74ff93b0b3ef468a713ffe73e53d1aec08f",
            "JobName": "my-testing-job",
            "StartedOn": 1602811020.518,
            "LastModifiedOn": 1602811138.364,
            "CompletedOn": 1602811138.364,
            "JobRunState": "FAILED",
            "ErrorMessage": "An error occurred while calling
 o122.pyWriteDynamicFrame.
                Access Denied (Service: Amazon S3; Status Code: 403; Error Code:
 AccessDenied;
                Request ID: 2671D37856AE7ABB;
                S3 Extended Request ID: RLJCJw20brV
+PpC6GpORahyF2fp9flB5SSb2bTGPnUSPVizLXRl1PN3QZldb+v1o9qRVktNYbW8=)",
            "PredecessorRuns": [],
            "AllocatedCapacity": 10,
            "ExecutionTime": 113,
            "Timeout": 2880,
            "MaxCapacity": 10.0,
            "WorkerType": "G.1X",
            "NumberOfWorkers": 10,
            "LogGroupName": "/aws-glue/jobs",
            "GlueVersion": "2.0"
        }
    ]
```

```
}
```

Para obtener más información, consulte Ejecuciones de trabajo en la Guía para desarrolladores de AWS Glue.

- Para obtener más información sobre la API, consulta GetJobRunsla Referencia de AWS CLI comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
const getJobRuns = (jobName) => {
  const client = new GlueClient({});
  const command = new GetJobRunsCommand({
    JobName: jobName,
  });

  return client.send(command);
};
```

- Para obtener más información sobre la API, consulta GetJobRunsla Referencia AWS SDK for JavaScript de la API.

PHP

SDK para PHP

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
        $jobName = 'test-job-' . $uniqid;

        $jobRuns = $glueService->getJobRuns($jobName);

    public function getJobRuns($jobName, $maxResults = 0, $nextToken = ''):
Result
    {
        $arguments = ['JobName' => $jobName];
        if ($maxResults) {
            $arguments['MaxResults'] = $maxResults;
        }
        if ($nextToken) {
            $arguments['NextToken'] = $nextToken;
        }
        return $this->glueClient->getJobRuns($arguments);
    }
```

- Para obtener más información sobre la API, consulta GetJobRunsla Referencia AWS SDK for PHP de la API.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client
```

```
    def get_job_runs(self, job_name):
        """
        Gets information about runs that have been performed for a specific job
        definition.

        :param job_name: The name of the job definition to look up.
        :return: The list of job runs.
        """
        try:
            response = self.glue_client.get_job_runs(JobName=job_name)
        except ClientError as err:
            logger.error(
                "Couldn't get job runs for %s. Here's why: %s: %s",
                job_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["JobRuns"]
```

- Para obtener más información sobre la API, consulta GetJobRunsla AWS Referencia de API de SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
```

```
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Retrieves a list of job runs for the specified job.
  #
  # @param job_name [String] The name of the job to retrieve job runs for.
  # @return [Array<Aws::Glue::Types::JobRun>]
  def get_job_runs(job_name)
    response = @glue_client.get_job_runs(job_name: job_name)
    response.job_runs
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get job runs: \n#{e.message}")
  end
```

- Para obtener más información sobre la API, consulta GetJobRunsla Referencia AWS SDK for Ruby de la API.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de este servicio con un AWS SDK. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`GetTables`** con un AWS SDK o CLI

En los siguientes ejemplos de código, se muestra cómo utilizar `GetTables`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- Comenzar a ejecutar rastreadores y trabajos

.NET

AWS SDK for .NET

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
/// <summary>
/// Get a list of tables for an AWS Glue database.
/// </summary>
/// <param name="dbName">The name of the database.</param>
/// <returns>A list of Table objects.</returns>
public async Task<List<Table>> GetTablesAsync(string dbName)
{
    var request = new GetTablesRequest { DatabaseName = dbName };
    var tables = new List<Table>();

    // Get a paginator for listing the tables.
    var tablePaginator = _amazonGlue.Paginators.GetTables(request);

    await foreach (var response in tablePaginator.Responses)
    {
        tables.AddRange(response.TableList);
    }

    return tables;
}
```

- Para obtener más información sobre la API, consulta GetTablesla Referencia AWS SDK for .NET de la API.

C++

SDK para C++

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
       Aws::Client::ClientConfiguration clientConfig;
       // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
       // clientConfig.region = "us-east-1";

   Aws::Glue::GlueClient client(clientConfig);

       Aws::Glue::Model::GetTablesRequest request;
       request.SetDatabaseName(CRAWLER_DATABASE_NAME);
       std::vector<Aws::Glue::Model::Table> all_tables;
       Aws::String nextToken; // Used for pagination.
       do {
           Aws::Glue::Model::GetTablesOutcome outcome =
client.GetTables(request);

           if (outcome.IsSuccess()) {
               const std::vector<Aws::Glue::Model::Table> &tables =
outcome.GetResult().GetTableList();
               all_tables.insert(all_tables.end(), tables.begin(),
tables.end());
               nextToken = outcome.GetResult().GetNextToken();
           }
           else {
               std::cerr << "Error getting the tables. "
                         << outcome.GetError().GetMessage()
                         << std::endl;
               deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                            clientConfig);
               return false;
           }
       } while (!nextToken.empty());
```

```
        std::cout << "The database contains " << all_tables.size()
                  << (all_tables.size() == 1 ?
                      " table." : "tables.") << std::endl;
        std::cout << "Here is a list of the tables in the database.";
        for (size_t index = 0; index < all_tables.size(); ++index) {
            std::cout << "    " << index + 1 << ":  " <<
all_tables[index].GetName()
                      << std::endl;
        }

        if (!all_tables.empty()) {
            int tableIndex = askQuestionForIntRange(
                    "Enter an index to display the database detail ",
                    1, static_cast<int>(all_tables.size()));
            std::cout << all_tables[tableIndex -
1].Jsonize().View().WriteReadable()
                      << std::endl;

            tableName = all_tables[tableIndex - 1].GetName();
        }
```

- Para obtener más información sobre la API, consulta GetTablesla Referencia AWS SDK for C++ de la API.

CLI

AWS CLI

Para enumerar las definiciones de algunas o todas las tablas de la base de datos especificada

El siguiente ejemplo de `get-tables` devuelve información sobre las tablas de la base de datos especificada.

```
aws glue get-tables --database-name 'tempdb'
```

Salida:

```
{
    "TableList": [
        {
            "Name": "my-s3-sink",
```

```
            "DatabaseName": "tempdb",
            "CreateTime": 1602730539.0,
            "UpdateTime": 1602730539.0,
            "Retention": 0,
            "StorageDescriptor": {
                "Columns": [
                    {
                        "Name": "sensorid",
                        "Type": "int"
                    },
                    {
                        "Name": "currenttemperature",
                        "Type": "int"
                    },
                    {
                        "Name": "status",
                        "Type": "string"
                    }
                ],
                "Location": "s3://janetst-bucket-01/test-s3-output/",
                "Compressed": false,
                "NumberOfBuckets": 0,
                "SerdeInfo": {
                    "SerializationLibrary": "org.openx.data.jsonserde.JsonSerDe"
                },
                "SortColumns": [],
                "StoredAsSubDirectories": false
            },
            "Parameters": {
                "classification": "json"
            },
            "CreatedBy": "arn:aws:iam::007436865787:user/JRSTERN",
            "IsRegisteredWithLakeFormation": false,
            "CatalogId": "007436865787"
        },
        {
            "Name": "s3-source",
            "DatabaseName": "tempdb",
            "CreateTime": 1602730658.0,
            "UpdateTime": 1602730658.0,
            "Retention": 0,
            "StorageDescriptor": {
                "Columns": [
                    {
```

```
                        "Name": "sensorid",
                        "Type": "int"
                    },
                    {
                        "Name": "currenttemperature",
                        "Type": "int"
                    },
                    {
                        "Name": "status",
                        "Type": "string"
                    }
                ],
                "Location": "s3://janetst-bucket-01/",
                "Compressed": false,
                "NumberOfBuckets": 0,
                "SortColumns": [],
                "StoredAsSubDirectories": false
            },
            "Parameters": {
                "classification": "json"
            },
            "CreatedBy": "arn:aws:iam::007436865787:user/JRSTERN",
            "IsRegisteredWithLakeFormation": false,
            "CatalogId": "007436865787"
        },
        {
            "Name": "test-kinesis-input",
            "DatabaseName": "tempdb",
            "CreateTime": 1601507001.0,
            "UpdateTime": 1601507001.0,
            "Retention": 0,
            "StorageDescriptor": {
                "Columns": [
                    {
                        "Name": "sensorid",
                        "Type": "int"
                    },
                    {
                        "Name": "currenttemperature",
                        "Type": "int"
                    },
                    {
                        "Name": "status",
                        "Type": "string"
```

```
                    }
                ],
                "Location": "my-testing-stream",
                "Compressed": false,
                "NumberOfBuckets": 0,
                "SerdeInfo": {
                    "SerializationLibrary": "org.openx.data.jsonserde.JsonSerDe"
                },
                "SortColumns": [],
                "Parameters": {
                    "kinesisUrl": "https://kinesis.us-east-1.amazonaws.com",
                    "streamName": "my-testing-stream",
                    "typeOfData": "kinesis"
                },
                "StoredAsSubDirectories": false
            },
            "Parameters": {
                "classification": "json"
            },
            "CreatedBy": "arn:aws:iam::007436865787:user/JRSTERN",
            "IsRegisteredWithLakeFormation": false,
            "CatalogId": "007436865787"
        }
    ]
}
```

Para obtener más información, consulte Definición de tablas en el catálogo de datos de AWS Glue en la Guía del desarrollador de AWS Glue.

- Para obtener más información sobre la API, consulte GetTablesla Referencia de AWS CLI comandos.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.GetTableRequest;
import software.amazon.awssdk.services.glue.model.GetTableResponse;
import software.amazon.awssdk.services.glue.model.GlueException;
import java.time.Instant;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;
import java.util.Locale;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class GetTable {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <dbName> <tableName>

                Where:
                    dbName - The database name.\s
                    tableName - The name of the table.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String dbName = args[0];
        String tableName = args[1];
        Region region = Region.US_EAST_1;
        GlueClient glueClient = GlueClient.builder()
                .region(region)
                .build();
```

```
        getGlueTable(glueClient, dbName, tableName);
        glueClient.close();
    }

    public static void getGlueTable(GlueClient glueClient, String dbName, String
 tableName) {
        try {
            GetTableRequest tableRequest = GetTableRequest.builder()
                    .databaseName(dbName)
                    .name(tableName)
                    .build();

            GetTableResponse tableResponse = glueClient.getTable(tableRequest);
            Instant createDate = tableResponse.table().createTime();

            // Convert the Instant to readable date.
            DateTimeFormatter formatter =
 DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
                    .withLocale(Locale.US)
                    .withZone(ZoneId.systemDefault());

            formatter.format(createDate);
            System.out.println("The create date of the table is " + createDate);

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obtener más información sobre la API, consulta GetTablesla Referencia AWS SDK for Java 2.x de la API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
const getTables = (databaseName) => {
  const client = new GlueClient({});

  const command = new GetTablesCommand({
    DatabaseName: databaseName,
  });

  return client.send(command);
};
```

- Para obtener más información sobre la API, consulta GetTablesla Referencia AWS SDK for JavaScript de la API.

PHP

SDK para PHP

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
        $databaseName = "doc-example-database-$uniqid";

        $tables = $glueService->getTables($databaseName);

    public function getTables($databaseName): Result
    {
```

```
        return $this->glueClient->getTables([
            'DatabaseName' => $databaseName,
        ]);
    }
```

- Para obtener más información sobre la API, consulta GetTablesla Referencia AWS SDK for PHP de la API.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def get_tables(self, db_name):
        """
        Gets a list of tables in a Data Catalog database.

        :param db_name: The name of the database to query.
        :return: The list of tables in the database.
        """
        try:
            response = self.glue_client.get_tables(DatabaseName=db_name)
        except ClientError as err:
            logger.error(
                "Couldn't get tables %s. Here's why: %s: %s",
```

```
                db_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["TableList"]
```

- Para obtener más información sobre la API, consulta GetTablesla AWS Referencia de API de SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Retrieves a list of tables in the specified database.
  #
  # @param db_name [String] The name of the database to retrieve tables from.
  # @return [Array<Aws::Glue::Types::Table>]
  def get_tables(db_name)
```

```
    response = @glue_client.get_tables(database_name: db_name)
    response.table_list
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get tables #{db_name}: \n#{e.message}")
    raise
  end
```

- Para obtener más información sobre la API, consulta GetTablesla Referencia AWS SDK for Ruby de la API.

Rust

SDK para Rust

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
        let tables = glue
            .get_tables()
            .database_name(self.database())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;

        let tables = tables.table_list();
```

- Para obtener más información sobre la API, consulta GetTablesla referencia sobre la API de AWS SDK para Rust.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de este servicio con un AWS SDK. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`ListJobs`** con un AWS SDK o CLI

En los siguientes ejemplos de código, se muestra cómo utilizar `ListJobs`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- Comenzar a ejecutar rastreadores y trabajos

.NET

AWS SDK for .NET

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    /// <summary>
    /// List AWS Glue jobs using a paginator.
    /// </summary>
    /// <returns>A list of AWS Glue job names.</returns>
    public async Task<List<string>> ListJobsAsync()
    {
        var jobNames = new List<string>();

        var listJobsPaginator = _amazonGlue.Paginators.ListJobs(new
ListJobsRequest { MaxResults = 10 });
        await foreach (var response in listJobsPaginator.Responses)
        {
            jobNames.AddRange(response.JobNames);
        }

        return jobNames;
    }
```

- Para obtener más información sobre la API, consulta ListJobsla Referencia AWS SDK for .NET de la API.

C++

SDK para C++

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
       Aws::Client::ClientConfiguration clientConfig;
       // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
       // clientConfig.region = "us-east-1";

   Aws::Glue::GlueClient client(clientConfig);

       Aws::Glue::Model::ListJobsRequest listJobsRequest;

       Aws::String nextToken;
       std::vector<Aws::String> allJobNames;

       do {
           if (!nextToken.empty()) {
               listJobsRequest.SetNextToken(nextToken);
           }
           Aws::Glue::Model::ListJobsOutcome listRunsOutcome = client.ListJobs(
                   listJobsRequest);

           if (listRunsOutcome.IsSuccess()) {
               const std::vector<Aws::String> &jobNames =
listRunsOutcome.GetResult().GetJobNames();
               allJobNames.insert(allJobNames.end(), jobNames.begin(),
jobNames.end());
               nextToken = listRunsOutcome.GetResult().GetNextToken();
           }
           else {
               std::cerr << "Error listing jobs. "
                         << listRunsOutcome.GetError().GetMessage()
                         << std::endl;
           }
       } while (!nextToken.empty());
```

- Para obtener más información sobre la API, consulta ListJobsla Referencia AWS SDK for C ++ de la API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
const listJobs = () => {
  const client = new GlueClient({});

  const command = new ListJobsCommand({});

  return client.send(command);
};
```

- Para obtener más información sobre la API, consulta ListJobsla Referencia AWS SDK for JavaScript de la API.

PHP

SDK para PHP

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
        $jobs = $glueService->listJobs();
        echo "Current jobs:\n";
```

```
        foreach ($jobs['JobNames'] as $jobsName) {
            echo "{$jobsName}\n";
        }

    public function listJobs($maxResults = null, $nextToken = null, $tags = []):
 Result
    {
        $arguments = [];
        if ($maxResults) {
            $arguments['MaxResults'] = $maxResults;
        }
        if ($nextToken) {
            $arguments['NextToken'] = $nextToken;
        }
        if (!empty($tags)) {
            $arguments['Tags'] = $tags;
        }
        return $this->glueClient->listJobs($arguments);
    }
```

- Para obtener más información sobre la API, consulta ListJobsla Referencia AWS SDK for PHP de la API.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
```

```
        self.glue_client = glue_client


    def list_jobs(self):
        """
        Lists the names of job definitions in your account.

        :return: The list of job definition names.
        """
        try:
            response = self.glue_client.list_jobs()
        except ClientError as err:
            logger.error(
                "Couldn't list jobs. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["JobNames"]
```

- Para obtener más información sobre la API, consulta ListJobsla AWS Referencia de API de SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
```

```
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Retrieves a list of jobs in AWS Glue.
  #
  # @return [Aws::Glue::Types::ListJobsResponse]
  def list_jobs
    @glue_client.list_jobs
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not list jobs: \n#{e.message}")
    raise
  end
```

- Para obtener más información sobre la API, consulta ListJobsla Referencia AWS SDK for Ruby de la API.

Rust

SDK para Rust

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
        let mut list_jobs = glue.list_jobs().into_paginator().send();
        while let Some(list_jobs_output) = list_jobs.next().await {
            match list_jobs_output {
                Ok(list_jobs) => {
                    let names = list_jobs.job_names();
                    info!(?names, "Found these jobs")
                }
                Err(err) => return Err(GlueMvpError::from_glue_sdk(err)),
            }
```

```
        }
```

- Para obtener más información sobre la API, consulta ListJobsla referencia sobre la API de AWS SDK para Rust.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de este servicio con un AWS SDK. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`StartCrawler`** con un AWS SDK o CLI

En los siguientes ejemplos de código, se muestra cómo utilizar `StartCrawler`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- Comenzar a ejecutar rastreadores y trabajos

.NET

AWS SDK for .NET

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
/// <summary>
/// Start an AWS Glue crawler.
/// </summary>
/// <param name="crawlerName">The name of the crawler.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> StartCrawlerAsync(string crawlerName)
{
    var crawlerRequest = new StartCrawlerRequest
    {
        Name = crawlerName,
    };
```

```
        var response = await _amazonGlue.StartCrawlerAsync(crawlerRequest);

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- Para obtener más información sobre la API, consulta StartCrawlerla Referencia AWS SDK for .NET de la API.

C++

SDK para C++

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
       Aws::Client::ClientConfiguration clientConfig;
       // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
       // clientConfig.region = "us-east-1";

   Aws::Glue::GlueClient client(clientConfig);

       Aws::Glue::Model::StartCrawlerRequest request;
       request.SetName(CRAWLER_NAME);

       Aws::Glue::Model::StartCrawlerOutcome outcome =
client.StartCrawler(request);


       if (outcome.IsSuccess() || (Aws::Glue::GlueErrors::CRAWLER_RUNNING ==
                                   outcome.GetError().GetErrorType())) {
           if (!outcome.IsSuccess()) {
               std::cout << "Crawler was already started." << std::endl;
           }
           else {
```

```
                std::cout << "Successfully started crawler." << std::endl;
            }

            std::cout << "This may take a while to run." << std::endl;

            Aws::Glue::Model::CrawlerState crawlerState =
Aws::Glue::Model::CrawlerState::NOT_SET;
            int iterations = 0;
            while (Aws::Glue::Model::CrawlerState::READY != crawlerState) {
                std::this_thread::sleep_for(std::chrono::seconds(1));
                ++iterations;
                if ((iterations % 10) == 0) { // Log status every 10 seconds.
                    std::cout << "Crawler status " <<

Aws::Glue::Model::CrawlerStateMapper::GetNameForCrawlerState(
                                        crawlerState)
                              << ". After " << iterations
                              << " seconds elapsed."
                              << std::endl;
                }
                Aws::Glue::Model::GetCrawlerRequest getCrawlerRequest;
                getCrawlerRequest.SetName(CRAWLER_NAME);

                Aws::Glue::Model::GetCrawlerOutcome getCrawlerOutcome =
client.GetCrawler(
                        getCrawlerRequest);

                if (getCrawlerOutcome.IsSuccess()) {
                    crawlerState =
getCrawlerOutcome.GetResult().GetCrawler().GetState();
                }
                else {
                    std::cerr << "Error getting crawler.  "
                              << getCrawlerOutcome.GetError().GetMessage() <<
std::endl;
                    break;
                }
            }

            if (Aws::Glue::Model::CrawlerState::READY == crawlerState) {
                std::cout << "Crawler finished running after " << iterations
                          << " seconds."
                          << std::endl;
            }
```

```
        }
        else {
            std::cerr << "Error starting a crawler.  "
                      << outcome.GetError().GetMessage()
                      << std::endl;

            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                         clientConfig);
            return false;
        }
```

- Para obtener más información sobre la API, consulta StartCrawlerla Referencia AWS SDK for C++ de la API.

CLI

AWS CLI

Para iniciar un rastreador

El siguiente ejemplo de `start-crawler` inicia un rastreador.

```
aws glue start-crawler --name my-crawler
```

Salida:

```
None
```

Para obtener más información, consulte Definición de rastreadores en la Guía para desarrolladores de AWS Glue.

- Para obtener más información sobre la API, consulta StartCrawlerla Referencia de AWS CLI comandos.

Java

SDK para Java 2.x

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.glue.GlueClient;
import software.amazon.awssdk.services.glue.model.GlueException;
import software.amazon.awssdk.services.glue.model.StartCrawlerRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class StartCrawler {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <crawlerName>

                Where:
                    crawlerName - The name of the crawler.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String crawlerName = args[0];
        Region region = Region.US_EAST_1;
```

```
        GlueClient glueClient = GlueClient.builder()
                .region(region)
                .build();

        startSpecificCrawler(glueClient, crawlerName);
        glueClient.close();
    }

    public static void startSpecificCrawler(GlueClient glueClient, String
 crawlerName) {
        try {
            StartCrawlerRequest crawlerRequest = StartCrawlerRequest.builder()
                    .name(crawlerName)
                    .build();

            glueClient.startCrawler(crawlerRequest);

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Para obtener más información sobre la API, consulta StartCrawlerla Referencia AWS SDK for Java 2.x de la API.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
const startCrawler = (name) => {
  const client = new GlueClient({});
```

```
  const command = new StartCrawlerCommand({
    Name: name,
  });

  return client.send(command);
};
```

- Para obtener más información sobre la API, consulta StartCrawlerla Referencia AWS SDK for JavaScript de la API.

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun startSpecificCrawler(crawlerName: String?) {
    val request =
        StartCrawlerRequest {
            name = crawlerName
        }

    GlueClient { region = "us-west-2" }.use { glueClient ->
        glueClient.startCrawler(request)
        println("$crawlerName was successfully started.")
    }
}
```

- Para obtener más información sobre la API, consulta StartCrawlerla referencia sobre el AWS SDK para la API de Kotlin.

PHP

SDK para PHP

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
        $crawlerName = "example-crawler-test-" . $uniqid;

        $databaseName = "doc-example-database-$uniqid";

        $glueService->startCrawler($crawlerName);

    public function startCrawler($crawlerName): Result
    {
        return $this->glueClient->startCrawler([
            'Name' => $crawlerName,
        ]);
    }
```

- Para obtener más información sobre la API, consulta StartCrawlerla Referencia AWS SDK for PHP de la API.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""
```

```
    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def start_crawler(self, name):
        """
        Starts a crawler. The crawler crawls its configured target and creates
        metadata that describes the data it finds in the target data source.

        :param name: The name of the crawler to start.
        """
        try:
            self.glue_client.start_crawler(Name=name)
        except ClientError as err:
            logger.error(
                "Couldn't start crawler %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- Para obtener más información sobre la API, consulta StartCrawlerla AWS Referencia de API de SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Starts a crawler with the specified name.
  #
  # @param name [String] The name of the crawler to start.
  # @return [void]
  def start_crawler(name)
    @glue_client.start_crawler(name: name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not start crawler #{name}: \n#{e.message}")
    raise
  end
```

- Para obtener más información sobre la API, consulta StartCrawlerla Referencia AWS SDK for Ruby de la API.

Rust

SDK para Rust

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
        let start_crawler =
glue.start_crawler().name(self.crawler()).send().await;

        match start_crawler {
```

```
            Ok(_) => Ok(()),
            Err(err) => {
                let glue_err: aws_sdk_glue::Error = err.into();
                match glue_err {
                    aws_sdk_glue::Error::CrawlerRunningException(_) => Ok(()),
                    _ => Err(GlueMvpError::GlueSdk(glue_err)),
                }
            }
        }?;
```

- Para obtener más información sobre la API, consulta StartCrawlerla referencia sobre la API de AWS SDK para Rust.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de este servicio con un AWS SDK. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Úselo **`StartJobRun`** con un AWS SDK o CLI

En los siguientes ejemplos de código, se muestra cómo utilizar `StartJobRun`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- Comenzar a ejecutar rastreadores y trabajos

.NET

AWS SDK for .NET

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
    /// <summary>
    /// Start an AWS Glue job run.
    /// </summary>
```

```
    /// <param name="jobName">The name of the job.</param>
    /// <returns>A string representing the job run Id.</returns>
    public async Task<string> StartJobRunAsync(
        string jobName,
        string inputDatabase,
        string inputTable,
        string bucketName)
    {
        var request = new StartJobRunRequest
        {
            JobName = jobName,
            Arguments = new Dictionary<string, string>
            {
                {"--input_database", inputDatabase},
                {"--input_table", inputTable},
                {"--output_bucket_url", $"s3://{bucketName}/"}
            }
        };

        var response = await _amazonGlue.StartJobRunAsync(request);
        return response.JobRunId;
    }
```

- Para obtener más información sobre la API, consulta StartJobRunla Referencia AWS SDK for .NET de la API.

C++

SDK para C++

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
```

```
        // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);

        Aws::Glue::Model::StartJobRunRequest request;
        request.SetJobName(JOB_NAME);

        Aws::Map<Aws::String, Aws::String> arguments;
        arguments["--input_database"] = CRAWLER_DATABASE_NAME;
        arguments["--input_table"] = tableName;
        arguments["--output_bucket_url"] = Aws::String("s3://") + bucketName +
"/";
        request.SetArguments(arguments);

        Aws::Glue::Model::StartJobRunOutcome outcome =
client.StartJobRun(request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully started the job." << std::endl;

            Aws::String jobRunId = outcome.GetResult().GetJobRunId();

            int iterator = 0;
            bool done = false;
            while (!done) {
                ++iterator;
                std::this_thread::sleep_for(std::chrono::seconds(1));
                Aws::Glue::Model::GetJobRunRequest jobRunRequest;
                jobRunRequest.SetJobName(JOB_NAME);
                jobRunRequest.SetRunId(jobRunId);

                Aws::Glue::Model::GetJobRunOutcome jobRunOutcome =
client.GetJobRun(
                        jobRunRequest);

                if (jobRunOutcome.IsSuccess()) {
                    const Aws::Glue::Model::JobRun &jobRun =
jobRunOutcome.GetResult().GetJobRun();
                    Aws::Glue::Model::JobRunState jobRunState =
jobRun.GetJobRunState();

                    if ((jobRunState == Aws::Glue::Model::JobRunState::STOPPED)
||
                        (jobRunState == Aws::Glue::Model::JobRunState::FAILED) ||
```

```
                        (jobRunState == Aws::Glue::Model::JobRunState::TIMEOUT))
{
                        std::cerr << "Error running job. "
                                  << jobRun.GetErrorMessage()
                                  << std::endl;
                        deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME,
JOB_NAME,
                                     bucketName,
                                     clientConfig);
                        return false;
                    }
                    else if (jobRunState ==
                             Aws::Glue::Model::JobRunState::SUCCEEDED) {
                        std::cout << "Job run succeeded after  " << iterator <<
                                  " seconds elapsed." << std::endl;
                        done = true;
                    }
                    else if ((iterator % 10) == 0) { // Log status every 10
seconds.
                        std::cout << "Job run status " <<

Aws::Glue::Model::JobRunStateMapper::GetNameForJobRunState(
                                         jobRunState) <<
                                  ". " << iterator <<
                                  " seconds elapsed." << std::endl;
                    }
                }
                else {
                    std::cerr << "Error retrieving job run state. "
                              << jobRunOutcome.GetError().GetMessage()
                              << std::endl;
                    deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME,
                                 bucketName, clientConfig);
                    return false;
                }
            }
        }
        else {
            std::cerr << "Error starting a job. " <<
outcome.GetError().GetMessage()
                      << std::endl;
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME,
bucketName,
                         clientConfig);
```

```
            return false;
        }
```

- Para obtener más información sobre la API, consulta StartJobRunla Referencia AWS SDK for C++ de la API.

CLI

AWS CLI

Para empezar a ejecutar un trabajo

El siguiente ejemplo de `start-job-run` inicia un trabajo.

```
aws glue start-job-run \
    --job-name my-job
```

Salida:

```
{
    "JobRunId":
 "jr_22208b1f44eb5376a60569d4b21dd20fcb8621e1a366b4e7b2494af764b82ded"
}
```

Para obtener más información, consulte Creación de trabajos en la Guía para desarrolladores de AWS Glue.

- Para obtener más información sobre la API, consulta StartJobRunla Referencia de AWS CLI comandos.

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
const startJobRun = (jobName, dbName, tableName, bucketName) => {
  const client = new GlueClient({});

  const command = new StartJobRunCommand({
    JobName: jobName,
    Arguments: {
      "--input_database": dbName,
      "--input_table": tableName,
      "--output_bucket_url": `s3://${bucketName}/`,
    },
  });

  return client.send(command);
};
```

- Para obtener más información sobre la API, consulta StartJobRunla Referencia AWS SDK for JavaScript de la API.

PHP

SDK para PHP

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
        $jobName = 'test-job-' . $uniqid;

        $databaseName = "doc-example-database-$uniqid";

        $tables = $glueService->getTables($databaseName);

        $outputBucketUrl = "s3://$bucketName";
        $runId = $glueService->startJobRun($jobName, $databaseName, $tables,
 $outputBucketUrl)['JobRunId'];

    public function startJobRun($jobName, $databaseName, $tables,
 $outputBucketUrl): Result
```

```
    {
        return $this->glueClient->startJobRun([
            'JobName' => $jobName,
            'Arguments' => [
                'input_database' => $databaseName,
                'input_table' => $tables['TableList'][0]['Name'],
                'output_bucket_url' => $outputBucketUrl,
                '--input_database' => $databaseName,
                '--input_table' => $tables['TableList'][0]['Name'],
                '--output_bucket_url' => $outputBucketUrl,
            ],
        ]);
    }
```

- Para obtener más información sobre la API, consulta StartJobRunla Referencia AWS SDK for PHP de la API.

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def start_job_run(self, name, input_database, input_table,
 output_bucket_name):
        """
```

```
        Starts a job run. A job run extracts data from the source, transforms it,
        and loads it to the output bucket.

        :param name: The name of the job definition.
        :param input_database: The name of the metadata database that contains
tables
                               that describe the source data. This is typically
created
                               by a crawler.
        :param input_table: The name of the table in the metadata database that
                            describes the source data.
        :param output_bucket_name: The S3 bucket where the output is written.
        :return: The ID of the job run.
        """
        try:
            # The custom Arguments that are passed to this function are used by
the
            # Python ETL script to determine the location of input and output
data.
            response = self.glue_client.start_job_run(
                JobName=name,
                Arguments={
                    "--input_database": input_database,
                    "--input_table": input_table,
                    "--output_bucket_url": f"s3://{output_bucket_name}/",
                },
            )
        except ClientError as err:
            logger.error(
                "Couldn't start job run %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["JobRunId"]
```

- Para obtener más información sobre la API, consulta StartJobRunla AWS Referencia de API de SDK for Python (Boto3).

Ruby

SDK para Ruby

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Starts a job run for the specified job.
  #
  # @param name [String] The name of the job to start the run for.
  # @param input_database [String] The name of the input database for the job.
  # @param input_table [String] The name of the input table for the job.
  # @param output_bucket_name [String] The name of the output S3 bucket for the
 job.
  # @return [String] The ID of the started job run.
  def start_job_run(name, input_database, input_table, output_bucket_name)
    response = @glue_client.start_job_run(
      job_name: name,
      arguments: {
        '--input_database': input_database,
        '--input_table': input_table,
        '--output_bucket_url': "s3://#{output_bucket_name}/"
      }
    )
    response.job_run_id
  rescue Aws::Glue::Errors::GlueException => e
```

```
    @logger.error("Glue could not start job run #{name}: \n#{e.message}")
    raise
  end
```

- Para obtener más información sobre la API, consulta StartJobRunla Referencia AWS SDK for Ruby de la API.

Rust

SDK para Rust

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
        let job_run_output = glue
            .start_job_run()
            .job_name(self.job())
            .arguments("--input_database", self.database())
            .arguments(
                "--input_table",
                self.tables
                    .first()
                    .ok_or_else(|| GlueMvpError::Unknown("Missing crawler
table".into()))?
                    .name(),
            )
            .arguments("--output_bucket_url", self.bucket())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;

        let job = job_run_output
            .job_run_id()
            .ok_or_else(|| GlueMvpError::Unknown("Missing run id from just
started job".into()))?
            .to_string();
```

- Para obtener más información sobre la API, consulta StartJobRunla referencia sobre la API de AWS SDK para Rust.

Para obtener una lista completa de guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de este servicio con un AWS SDK. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Escenarios de AWS Glue uso de AWS los SDK

Los siguientes ejemplos de código muestran cómo implementar escenarios comunes AWS Glue con los AWS SDK. Estos escenarios muestran cómo realizar tareas específicas mediante la invocación de varias funciones internas AWS Glue. Cada escenario incluye un enlace a GitHub, donde puede encontrar instrucciones sobre cómo configurar y ejecutar el código.

Ejemplos

- Comience a ejecutar AWS Glue rastreadores y tareas con un AWS SDK

Comience a ejecutar AWS Glue rastreadores y tareas con un AWS SDK

En el siguiente ejemplo de código, se muestra cómo:

- Cree un rastreador que rastree un bucket de Amazon S3 público y genere una base de datos de metadatos con formato CSV.
- Enumere información sobre bases de datos y tablas en su AWS Glue Data Catalog.
- Crear un trabajo para extraer datos CSV del bucket de S3, transformar los datos y cargar el resultado con formato JSON en otro bucket de S3.
- Incluir información sobre las ejecuciones de trabajos, ver algunos de los datos transformados y limpiar los recursos.

Para obtener más información, consulta el tutorial: Cómo empezar a usar AWS Glue Studio.

.NET

AWS SDK for .NET

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Cree una clase que agrupe AWS Glue las funciones que se utilizan en el escenario.

```
using System.Net;

namespace GlueActions;

public class GlueWrapper
{
    private readonly IAmazonGlue _amazonGlue;

    /// <summary>
    /// Constructor for the AWS Glue actions wrapper.
    /// </summary>
    /// <param name="amazonGlue"></param>
    public GlueWrapper(IAmazonGlue amazonGlue)
    {
        _amazonGlue = amazonGlue;
    }

    /// <summary>
    /// Create an AWS Glue crawler.
    /// </summary>
    /// <param name="crawlerName">The name for the crawler.</param>
    /// <param name="crawlerDescription">A description of the crawler.</param>
    /// <param name="role">The AWS Identity and Access Management (IAM) role to
    /// be assumed by the crawler.</param>
    /// <param name="schedule">The schedule on which the crawler will be
 executed.</param>
    /// <param name="s3Path">The path to the Amazon Simple Storage Service
 (Amazon S3)
    /// bucket where the Python script has been stored.</param>
    /// <param name="dbName">The name to use for the database that will be
```

```
    /// created by the crawler.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> CreateCrawlerAsync(
        string crawlerName,
        string crawlerDescription,
        string role,
        string schedule,
        string s3Path,
        string dbName)
    {
        var s3Target = new S3Target
        {
            Path = s3Path,
        };

        var targetList = new List<S3Target>
        {
            s3Target,
        };

        var targets = new CrawlerTargets
        {
            S3Targets = targetList,
        };

        var crawlerRequest = new CreateCrawlerRequest
        {
            DatabaseName = dbName,
            Name = crawlerName,
            Description = crawlerDescription,
            Targets = targets,
            Role = role,
            Schedule = schedule,
        };

        var response = await _amazonGlue.CreateCrawlerAsync(crawlerRequest);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Create an AWS Glue job.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
```

```
    /// <param name="roleName">The name of the IAM role to be assumed by
    /// the job.</param>
    /// <param name="description">A description of the job.</param>
    /// <param name="scriptUrl">The URL to the script.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> CreateJobAsync(string dbName, string tableName,
string bucketUrl, string jobName, string roleName, string description, string
scriptUrl)
    {
        var command = new JobCommand
        {
            PythonVersion = "3",
            Name = "glueetl",
            ScriptLocation = scriptUrl,
        };

        var arguments = new Dictionary<string, string>
        {
            { "--input_database", dbName },
            { "--input_table", tableName },
            { "--output_bucket_url", bucketUrl }
        };

        var request = new CreateJobRequest
        {
            Command = command,
            DefaultArguments = arguments,
            Description = description,
            GlueVersion = "3.0",
            Name = jobName,
            NumberOfWorkers = 10,
            Role = roleName,
            WorkerType = "G.1X"
        };

        var response = await _amazonGlue.CreateJobAsync(request);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an AWS Glue crawler.
    /// </summary>
    /// <param name="crawlerName">The name of the crawler.</param>
```

```
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteCrawlerAsync(string crawlerName)
    {
        var response = await _amazonGlue.DeleteCrawlerAsync(new
DeleteCrawlerRequest { Name = crawlerName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete the AWS Glue database.
    /// </summary>
    /// <param name="dbName">The name of the database.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteDatabaseAsync(string dbName)
    {
        var response = await _amazonGlue.DeleteDatabaseAsync(new
DeleteDatabaseRequest { Name = dbName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an AWS Glue job.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteJobAsync(string jobName)
    {
        var response = await _amazonGlue.DeleteJobAsync(new DeleteJobRequest
{ JobName = jobName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete a table from an AWS Glue database.
    /// </summary>
    /// <param name="tableName">The table to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteTableAsync(string dbName, string tableName)
    {
        var response = await _amazonGlue.DeleteTableAsync(new DeleteTableRequest
{ Name = tableName, DatabaseName = dbName });
```

```
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Get information about an AWS Glue crawler.
    /// </summary>
    /// <param name="crawlerName">The name of the crawler.</param>
    /// <returns>A Crawler object describing the crawler.</returns>
    public async Task<Crawler?> GetCrawlerAsync(string crawlerName)
    {
        var crawlerRequest = new GetCrawlerRequest
        {
            Name = crawlerName,
        };

        var response = await _amazonGlue.GetCrawlerAsync(crawlerRequest);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            var databaseName = response.Crawler.DatabaseName;
            Console.WriteLine($"{crawlerName} has the database {databaseName}");
            return response.Crawler;
        }

        Console.WriteLine($"No information regarding {crawlerName} could be
found.");
        return null;
    }


    /// <summary>
    /// Get information about the state of an AWS Glue crawler.
    /// </summary>
    /// <param name="crawlerName">The name of the crawler.</param>
    /// <returns>A value describing the state of the crawler.</returns>
    public async Task<CrawlerState> GetCrawlerStateAsync(string crawlerName)
    {
        var response = await _amazonGlue.GetCrawlerAsync(
            new GetCrawlerRequest { Name = crawlerName });
        return response.Crawler.State;
    }


    /// <summary>
```

```
    /// Get information about an AWS Glue database.
    /// </summary>
    /// <param name="dbName">The name of the database.</param>
    /// <returns>A Database object containing information about the database.</
returns>
    public async Task<Database> GetDatabaseAsync(string dbName)
    {
        var databasesRequest = new GetDatabaseRequest
        {
            Name = dbName,
        };

        var response = await _amazonGlue.GetDatabaseAsync(databasesRequest);
        return response.Database;
    }


    /// <summary>
    /// Get information about a specific AWS Glue job run.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <param name="jobRunId">The Id of the job run.</param>
    /// <returns>A JobRun object with information about the job run.</returns>
    public async Task<JobRun> GetJobRunAsync(string jobName, string jobRunId)
    {
        var response = await _amazonGlue.GetJobRunAsync(new GetJobRunRequest
 { JobName = jobName, RunId = jobRunId });
        return response.JobRun;
    }


    /// <summary>
    /// Get information about all AWS Glue runs of a specific job.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <returns>A list of JobRun objects.</returns>
    public async Task<List<JobRun>> GetJobRunsAsync(string jobName)
    {
        var jobRuns = new List<JobRun>();

        var request = new GetJobRunsRequest
        {
            JobName = jobName,
        };
```

```
        // No need to loop to get all the log groups--the SDK does it for us
behind the scenes
        var paginatorForJobRuns =
            _amazonGlue.Paginators.GetJobRuns(request);

        await foreach (var response in paginatorForJobRuns.Responses)
        {
            response.JobRuns.ForEach(jobRun =>
            {
                jobRuns.Add(jobRun);
            });
        }

        return jobRuns;
    }


    /// <summary>
    /// Get a list of tables for an AWS Glue database.
    /// </summary>
    /// <param name="dbName">The name of the database.</param>
    /// <returns>A list of Table objects.</returns>
    public async Task<List<Table>> GetTablesAsync(string dbName)
    {
        var request = new GetTablesRequest { DatabaseName = dbName };
        var tables = new List<Table>();

        // Get a paginator for listing the tables.
        var tablePaginator = _amazonGlue.Paginators.GetTables(request);

        await foreach (var response in tablePaginator.Responses)
        {
            tables.AddRange(response.TableList);
        }

        return tables;
    }


    /// <summary>
    /// List AWS Glue jobs using a paginator.
    /// </summary>
    /// <returns>A list of AWS Glue job names.</returns>
```

```
    public async Task<List<string>> ListJobsAsync()
    {
        var jobNames = new List<string>();

        var listJobsPaginator = _amazonGlue.Paginators.ListJobs(new
ListJobsRequest { MaxResults = 10 });
        await foreach (var response in listJobsPaginator.Responses)
        {
            jobNames.AddRange(response.JobNames);
        }

        return jobNames;
    }


    /// <summary>
    /// Start an AWS Glue crawler.
    /// </summary>
    /// <param name="crawlerName">The name of the crawler.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> StartCrawlerAsync(string crawlerName)
    {
        var crawlerRequest = new StartCrawlerRequest
        {
            Name = crawlerName,
        };

        var response = await _amazonGlue.StartCrawlerAsync(crawlerRequest);

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Start an AWS Glue job run.
    /// </summary>
    /// <param name="jobName">The name of the job.</param>
    /// <returns>A string representing the job run Id.</returns>
    public async Task<string> StartJobRunAsync(
        string jobName,
        string inputDatabase,
        string inputTable,
        string bucketName)
    {
```

```
        var request = new StartJobRunRequest
        {
            JobName = jobName,
            Arguments = new Dictionary<string, string>
            {
                {"--input_database", inputDatabase},
                {"--input_table", inputTable},
                {"--output_bucket_url", $"s3://{bucketName}/"}
            }
        };

        var response = await _amazonGlue.StartJobRunAsync(request);
        return response.JobRunId;
    }

}
```

Cree una clase que ejecute el escenario.

```
global using Amazon.Glue;
global using GlueActions;
global using Microsoft.Extensions.Configuration;
global using Microsoft.Extensions.DependencyInjection;
global using Microsoft.Extensions.Hosting;
global using Microsoft.Extensions.Logging;
global using Microsoft.Extensions.Logging.Console;
global using Microsoft.Extensions.Logging.Debug;



using Amazon.Glue.Model;
using Amazon.S3;
using Amazon.S3.Model;

namespace GlueBasics;

public class GlueBasics
{
    private static ILogger logger = null!;
    private static IConfiguration _configuration = null!;
```

```
   static async Task Main(string[] args)
   {
       // Set up dependency injection for AWS Glue.
       using var host = Host.CreateDefaultBuilder(args)
           .ConfigureLogging(logging =>
               logging.AddFilter("System", LogLevel.Debug)
                   .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                   .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
           .ConfigureServices((_, services) =>
           services.AddAWSService<IAmazonGlue>()
           .AddTransient<GlueWrapper>()
           .AddTransient<UiWrapper>()
           )
           .Build();

       logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
       .CreateLogger<GlueBasics>();

       _configuration = new ConfigurationBuilder()
           .SetBasePath(Directory.GetCurrentDirectory())
           .AddJsonFile("settings.json") // Load settings from .json file.
           .AddJsonFile("settings.local.json",
               true) // Optionally load local settings.
           .Build();

       // These values are stored in settings.json
       // Once you have run the CDK script to deploy the resources,
       // edit the file to set "BucketName", "RoleName", and "ScriptURL"
       // to the appropriate values. Also set "CrawlerName" to the name
       // you want to give the crawler when it is created.
       string bucketName = _configuration["BucketName"]!;
       string bucketUrl = _configuration["BucketUrl"]!;
       string crawlerName = _configuration["CrawlerName"]!;
       string roleName = _configuration["RoleName"]!;
       string sourceData = _configuration["SourceData"]!;
       string dbName = _configuration["DbName"]!;
       string cron = _configuration["Cron"]!;
       string scriptUrl = _configuration["ScriptURL"]!;
       string jobName = _configuration["JobName"]!;

       var wrapper = host.Services.GetRequiredService<GlueWrapper>();
       var uiWrapper = host.Services.GetRequiredService<UiWrapper>();
```

```
        uiWrapper.DisplayOverview();
        uiWrapper.PressEnter();

        // Create the crawler and wait for it to be ready.
        uiWrapper.DisplayTitle("Create AWS Glue crawler");
        Console.WriteLine("Let's begin by creating the AWS Glue crawler.");

        var crawlerDescription = "Crawler created for the AWS Glue Basics
scenario.";
        var crawlerCreated = await wrapper.CreateCrawlerAsync(crawlerName,
crawlerDescription, roleName, cron, sourceData, dbName);
        if (crawlerCreated)
        {
            Console.WriteLine($"The crawler: {crawlerName} has been created. Now
let's wait until it's ready.");
            CrawlerState crawlerState;
            do
            {
                crawlerState = await wrapper.GetCrawlerStateAsync(crawlerName);
            }
            while (crawlerState != "READY");
            Console.WriteLine($"The crawler {crawlerName} is now ready for
use.");
        }
        else
        {
            Console.WriteLine($"Couldn't create crawler {crawlerName}.");
            return; // Exit the application.
        }

        uiWrapper.DisplayTitle("Start AWS Glue crawler");
        Console.WriteLine("Now let's wait until the crawler has successfully
started.");
        var crawlerStarted = await wrapper.StartCrawlerAsync(crawlerName);
        if (crawlerStarted)
        {
            CrawlerState crawlerState;
            do
            {
                crawlerState = await wrapper.GetCrawlerStateAsync(crawlerName);
            }
            while (crawlerState != "READY");
```

```
            Console.WriteLine($"The crawler {crawlerName} is now ready for
use.");
        }
        else
        {
            Console.WriteLine($"Couldn't start the crawler {crawlerName}.");
            return; // Exit the application.
        }

        uiWrapper.PressEnter();

        Console.WriteLine($"\nLet's take a look at the database: {dbName}");
        var database = await wrapper.GetDatabaseAsync(dbName);

        if (database != null)
        {
            uiWrapper.DisplayTitle($"{database.Name} Details");
            Console.WriteLine($"{database.Name} created on
{database.CreateTime}");
            Console.WriteLine(database.Description);
        }

        uiWrapper.PressEnter();

        var tables = await wrapper.GetTablesAsync(dbName);
        if (tables.Count > 0)
        {
            tables.ForEach(table =>
            {
                Console.WriteLine($"{table.Name}\tCreated:
{table.CreateTime}\tUpdated: {table.UpdateTime}");
            });
        }

        uiWrapper.PressEnter();

        uiWrapper.DisplayTitle("Create AWS Glue job");
        Console.WriteLine("Creating a new AWS Glue job.");
        var description = "An AWS Glue job created using the AWS SDK for .NET";
        await wrapper.CreateJobAsync(dbName, tables[0].Name, bucketUrl, jobName,
roleName, description, scriptUrl);

        uiWrapper.PressEnter();
```

```
        uiWrapper.DisplayTitle("Starting AWS Glue job");
        Console.WriteLine("Starting the new AWS Glue job...");
        var jobRunId = await wrapper.StartJobRunAsync(jobName, dbName,
tables[0].Name, bucketName);
        var jobRunComplete = false;
        var jobRun = new JobRun();
        do
        {
            jobRun = await wrapper.GetJobRunAsync(jobName, jobRunId);
            if (jobRun.JobRunState == "SUCCEEDED" || jobRun.JobRunState ==
"STOPPED" ||
                jobRun.JobRunState == "FAILED" || jobRun.JobRunState ==
"TIMEOUT")
            {
                jobRunComplete = true;
            }
        } while (!jobRunComplete);

        uiWrapper.DisplayTitle($"Data in {bucketName}");

        // Get the list of data stored in the S3 bucket.
        var s3Client = new AmazonS3Client();

        var response = await s3Client.ListObjectsAsync(new ListObjectsRequest
{ BucketName = bucketName });
        response.S3Objects.ForEach(s3Object =>
        {
            Console.WriteLine(s3Object.Key);
        });

        uiWrapper.DisplayTitle("AWS Glue jobs");
        var jobNames = await wrapper.ListJobsAsync();
        jobNames.ForEach(jobName =>
        {
            Console.WriteLine(jobName);
        });

        uiWrapper.PressEnter();

        uiWrapper.DisplayTitle("Get AWS Glue job run information");
        Console.WriteLine("Getting information about the AWS Glue job.");
        var jobRuns = await wrapper.GetJobRunsAsync(jobName);

        jobRuns.ForEach(jobRun =>
```

```
        {

 Console.WriteLine($"{jobRun.JobName}\t{jobRun.JobRunState}\t{jobRun.CompletedOn}");
        });

        uiWrapper.PressEnter();

        uiWrapper.DisplayTitle("Deleting resources");
        Console.WriteLine("Deleting the AWS Glue job used by the example.");
        await wrapper.DeleteJobAsync(jobName);

        Console.WriteLine("Deleting the tables from the database.");
        tables.ForEach(async table =>
        {
            await wrapper.DeleteTableAsync(dbName, table.Name);
        });

        Console.WriteLine("Deleting the database.");
        await wrapper.DeleteDatabaseAsync(dbName);

        Console.WriteLine("Deleting the AWS Glue crawler.");
        await wrapper.DeleteCrawlerAsync(crawlerName);

        Console.WriteLine("The AWS Glue scenario has completed.");
        uiWrapper.PressEnter();
    }
}


namespace GlueBasics;

public class UiWrapper
{
    public readonly string SepBar = new string('-', Console.WindowWidth);

    /// <summary>
    /// Show information about the scenario.
    /// </summary>
    public void DisplayOverview()
    {
        Console.Clear();
        DisplayTitle("Amazon Glue: get started with crawlers and jobs");

        Console.WriteLine("This example application does the following:");
```

```
        Console.WriteLine("\t 1. Create a crawler, pass it the IAM role and the
URL to the public S3 bucket that contains the source data");
        Console.WriteLine("\t 2. Start the crawler.");
        Console.WriteLine("\t 3. Get the database created by the crawler and the
tables in the database.");
        Console.WriteLine("\t 4. Create a job.");
        Console.WriteLine("\t 5. Start a job run.");
        Console.WriteLine("\t 6. Wait for the job run to complete.");
        Console.WriteLine("\t 7. Show the data stored in the bucket.");
        Console.WriteLine("\t 8. List jobs for the account.");
        Console.WriteLine("\t 9. Get job run details for the job that was run.");
        Console.WriteLine("\t10. Delete the demo job.");
        Console.WriteLine("\t11. Delete the database and tables created for the
demo.");
        Console.WriteLine("\t12. Delete the crawler.");
    }

    /// <summary>
    /// Display a message and wait until the user presses enter.
    /// </summary>
    public void PressEnter()
    {
        Console.Write("\nPlease press <Enter> to continue. ");
        _ = Console.ReadLine();
    }

    /// <summary>
    /// Pad a string with spaces to center it on the console display.
    /// </summary>
    /// <param name="strToCenter">The string to center on the screen.</param>
    /// <returns>The string padded to make it center on the screen.</returns>
    public string CenterString(string strToCenter)
    {
        var padAmount = (Console.WindowWidth - strToCenter.Length) / 2;
        var leftPad = new string(' ', padAmount);
        return $"{leftPad}{strToCenter}";
    }

    /// <summary>
    /// Display a line of hyphens, the centered text of the title and another
    /// line of hyphens.
    /// </summary>
    /// <param name="strTitle">The string to be displayed.</param>
    public void DisplayTitle(string strTitle)
```

```
    {
        Console.WriteLine(SepBar);
        Console.WriteLine(CenterString(strTitle));
        Console.WriteLine(SepBar);
    }
}
```

- Para obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK for .NET .
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun
  - GetJobRuns
  - GetTables
  - ListJobs
  - StartCrawler
  - StartJobRun

C++

SDK para C++

> **Note**
>
> Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
//! Scenario which demonstrates using AWS Glue to add a crawler and run a job.
/*!
 \\sa runGettingStartedWithGlueScenario()
 \param bucketName: An S3 bucket created in the setup.
 \param roleName: An AWS Identity and Access Management (IAM) role created in the
 setup.
 \param clientConfig: AWS client configuration.
 \return bool: Successful completion.
 */

bool AwsDoc::Glue::runGettingStartedWithGlueScenario(const Aws::String
 &bucketName,
                                                     const Aws::String &roleName,
                                                     const
 Aws::Client::ClientConfiguration &clientConfig) {
    Aws::Glue::GlueClient client(clientConfig);

    Aws::String roleArn;
    if (!getRoleArn(roleName, roleArn, clientConfig)) {
        std::cerr << "Error getting role ARN for role." << std::endl;
        return false;
    }

    // 1. Upload the job script to the S3 bucket.
    {
        std::cout << "Uploading the job script '"
                  << AwsDoc::Glue::PYTHON_SCRIPT
                  << "'." << std::endl;

        if (!AwsDoc::Glue::uploadFile(bucketName,
                                      AwsDoc::Glue::PYTHON_SCRIPT_PATH,
                                      AwsDoc::Glue::PYTHON_SCRIPT,
```

```
                                        clientConfig)) {
            std::cerr << "Error uploading the job file." << std::endl;
            return false;
        }
    }

    // 2. Create a crawler.
    {
        Aws::Glue::Model::S3Target s3Target;
        s3Target.SetPath("s3://crawler-public-us-east-1/flight/2016/csv");
        Aws::Glue::Model::CrawlerTargets crawlerTargets;
        crawlerTargets.AddS3Targets(s3Target);

        Aws::Glue::Model::CreateCrawlerRequest request;
        request.SetTargets(crawlerTargets);
        request.SetName(CRAWLER_NAME);
        request.SetDatabaseName(CRAWLER_DATABASE_NAME);
        request.SetTablePrefix(CRAWLER_DATABASE_PREFIX);
        request.SetRole(roleArn);

        Aws::Glue::Model::CreateCrawlerOutcome outcome =
client.CreateCrawler(request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully created the crawler." << std::endl;
        }
        else {
            std::cerr << "Error creating a crawler. " <<
outcome.GetError().GetMessage()
                      << std::endl;
            deleteAssets("", CRAWLER_DATABASE_NAME, "", bucketName,
clientConfig);
            return false;
        }
    }

    // 3. Get a crawler.
    {
        Aws::Glue::Model::GetCrawlerRequest request;
        request.SetName(CRAWLER_NAME);

        Aws::Glue::Model::GetCrawlerOutcome outcome = client.GetCrawler(request);

        if (outcome.IsSuccess()) {
```

```
            Aws::Glue::Model::CrawlerState crawlerState =
outcome.GetResult().GetCrawler().GetState();
            std::cout << "Retrieved crawler with state " <<

Aws::Glue::Model::CrawlerStateMapper::GetNameForCrawlerState(
                              crawlerState)
                      << "." << std::endl;
        }
        else {
            std::cerr << "Error retrieving a crawler.  "
                      << outcome.GetError().GetMessage() << std::endl;
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                         clientConfig);
            return false;
        }
    }

    // 4. Start a crawler.
    {
        Aws::Glue::Model::StartCrawlerRequest request;
        request.SetName(CRAWLER_NAME);

        Aws::Glue::Model::StartCrawlerOutcome outcome =
client.StartCrawler(request);


        if (outcome.IsSuccess() || (Aws::Glue::GlueErrors::CRAWLER_RUNNING ==
                                    outcome.GetError().GetErrorType())) {
            if (!outcome.IsSuccess()) {
                std::cout << "Crawler was already started." << std::endl;
            }
            else {
                std::cout << "Successfully started crawler." << std::endl;
            }

            std::cout << "This may take a while to run." << std::endl;

            Aws::Glue::Model::CrawlerState crawlerState =
Aws::Glue::Model::CrawlerState::NOT_SET;
            int iterations = 0;
            while (Aws::Glue::Model::CrawlerState::READY != crawlerState) {
                std::this_thread::sleep_for(std::chrono::seconds(1));
                ++iterations;
                if ((iterations % 10) == 0) { // Log status every 10 seconds.
```

```
                    std::cout << "Crawler status " <<

Aws::Glue::Model::CrawlerStateMapper::GetNameForCrawlerState(
                                      crawlerState)
                              << ". After " << iterations
                              << " seconds elapsed."
                              << std::endl;
                }
                Aws::Glue::Model::GetCrawlerRequest getCrawlerRequest;
                getCrawlerRequest.SetName(CRAWLER_NAME);

                Aws::Glue::Model::GetCrawlerOutcome getCrawlerOutcome =
client.GetCrawler(
                        getCrawlerRequest);

                if (getCrawlerOutcome.IsSuccess()) {
                    crawlerState =
getCrawlerOutcome.GetResult().GetCrawler().GetState();
                }
                else {
                    std::cerr << "Error getting crawler.  "
                              << getCrawlerOutcome.GetError().GetMessage() <<
std::endl;
                    break;
                }
            }

            if (Aws::Glue::Model::CrawlerState::READY == crawlerState) {
                std::cout << "Crawler finished running after " << iterations
                          << " seconds."
                          << std::endl;
            }
        }
        else {
            std::cerr << "Error starting a crawler.  "
                      << outcome.GetError().GetMessage()
                      << std::endl;

            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                         clientConfig);
            return false;
        }
    }
```

```
   // 5. Get a database.
   {
       Aws::Glue::Model::GetDatabaseRequest request;
       request.SetName(CRAWLER_DATABASE_NAME);

       Aws::Glue::Model::GetDatabaseOutcome outcome =
client.GetDatabase(request);

       if (outcome.IsSuccess()) {
           const Aws::Glue::Model::Database &database =
outcome.GetResult().GetDatabase();

           std::cout << "Successfully retrieve the database\n" <<
                     database.Jsonize().View().WriteReadable() << "'." <<
std::endl;
       }
       else {
           std::cerr << "Error getting the database.  "
                     << outcome.GetError().GetMessage() << std::endl;
           deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                        clientConfig);
           return false;
       }
   }

   // 6. Get tables.
   Aws::String tableName;
   {
       Aws::Glue::Model::GetTablesRequest request;
       request.SetDatabaseName(CRAWLER_DATABASE_NAME);
       std::vector<Aws::Glue::Model::Table> all_tables;
       Aws::String nextToken; // Used for pagination.
       do {
           Aws::Glue::Model::GetTablesOutcome outcome =
client.GetTables(request);

           if (outcome.IsSuccess()) {
               const std::vector<Aws::Glue::Model::Table> &tables =
outcome.GetResult().GetTableList();
               all_tables.insert(all_tables.end(), tables.begin(),
tables.end());
               nextToken = outcome.GetResult().GetNextToken();
           }
           else {
```

```
                std::cerr << "Error getting the tables. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                             clientConfig);
                return false;
            }
        } while (!nextToken.empty());

        std::cout << "The database contains " << all_tables.size()
                  << (all_tables.size() == 1 ?
                      " table." : "tables.") << std::endl;
        std::cout << "Here is a list of the tables in the database.";
        for (size_t index = 0; index < all_tables.size(); ++index) {
            std::cout << "    " << index + 1 << ":  " <<
all_tables[index].GetName()
                      << std::endl;
        }

        if (!all_tables.empty()) {
            int tableIndex = askQuestionForIntRange(
                    "Enter an index to display the database detail ",
                    1, static_cast<int>(all_tables.size()));
            std::cout << all_tables[tableIndex -
1].Jsonize().View().WriteReadable()
                      << std::endl;

            tableName = all_tables[tableIndex - 1].GetName();
        }
    }

    // 7. Create a job.
    {
        Aws::Glue::Model::CreateJobRequest request;
        request.SetName(JOB_NAME);
        request.SetRole(roleArn);
        request.SetGlueVersion(GLUE_VERSION);

        Aws::Glue::Model::JobCommand command;
        command.SetName(JOB_COMMAND_NAME);
        command.SetPythonVersion(JOB_PYTHON_VERSION);
        command.SetScriptLocation(
                Aws::String("s3://") + bucketName + "/" + PYTHON_SCRIPT);
        request.SetCommand(command);
```

```
        Aws::Glue::Model::CreateJobOutcome outcome = client.CreateJob(request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully created the job." << std::endl;
        }
        else {
            std::cerr << "Error creating the job. " <<
outcome.GetError().GetMessage()
                      << std::endl;
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                         clientConfig);
            return false;
        }
    }

    // 8. Start a job run.
    {
        Aws::Glue::Model::StartJobRunRequest request;
        request.SetJobName(JOB_NAME);

        Aws::Map<Aws::String, Aws::String> arguments;
        arguments["--input_database"] = CRAWLER_DATABASE_NAME;
        arguments["--input_table"] = tableName;
        arguments["--output_bucket_url"] = Aws::String("s3://") + bucketName +
"/";
        request.SetArguments(arguments);

        Aws::Glue::Model::StartJobRunOutcome outcome =
client.StartJobRun(request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully started the job." << std::endl;

            Aws::String jobRunId = outcome.GetResult().GetJobRunId();

            int iterator = 0;
            bool done = false;
            while (!done) {
                ++iterator;
                std::this_thread::sleep_for(std::chrono::seconds(1));
                Aws::Glue::Model::GetJobRunRequest jobRunRequest;
                jobRunRequest.SetJobName(JOB_NAME);
                jobRunRequest.SetRunId(jobRunId);
```

```
                Aws::Glue::Model::GetJobRunOutcome jobRunOutcome =
client.GetJobRun(
                        jobRunRequest);

                if (jobRunOutcome.IsSuccess()) {
                    const Aws::Glue::Model::JobRun &jobRun =
jobRunOutcome.GetResult().GetJobRun();
                    Aws::Glue::Model::JobRunState jobRunState =
jobRun.GetJobRunState();

                    if ((jobRunState == Aws::Glue::Model::JobRunState::STOPPED)
||
                        (jobRunState == Aws::Glue::Model::JobRunState::FAILED) ||
                        (jobRunState == Aws::Glue::Model::JobRunState::TIMEOUT))
{
                        std::cerr << "Error running job. "
                                  << jobRun.GetErrorMessage()
                                  << std::endl;
                        deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME,
JOB_NAME,
                                     bucketName,
                                     clientConfig);
                        return false;
                    }
                    else if (jobRunState ==
                             Aws::Glue::Model::JobRunState::SUCCEEDED) {
                        std::cout << "Job run succeeded after  " << iterator <<
                                  " seconds elapsed." << std::endl;
                        done = true;
                    }
                    else if ((iterator % 10) == 0) { // Log status every 10
seconds.
                        std::cout << "Job run status " <<

Aws::Glue::Model::JobRunStateMapper::GetNameForJobRunState(
                                         jobRunState) <<
                                  ". " << iterator <<
                                  " seconds elapsed." << std::endl;
                    }
                }
                else {
                    std::cerr << "Error retrieving job run state. "
                              << jobRunOutcome.GetError().GetMessage()
```

```
                            << std::endl;
                    deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME,
                                 bucketName, clientConfig);
                    return false;
                }
            }
        }
        else {
            std::cerr << "Error starting a job. " <<
outcome.GetError().GetMessage()
                      << std::endl;
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME,
bucketName,
                         clientConfig);
            return false;
        }
    }

    // 9. List the output data stored in the S3 bucket.
    {
        Aws::S3::S3Client s3Client;
        Aws::S3::Model::ListObjectsV2Request request;
        request.SetBucket(bucketName);
        request.SetPrefix(OUTPUT_FILE_PREFIX);

        Aws::String continuationToken; // Used for pagination.
        std::vector<Aws::S3::Model::Object> allObjects;
        do {
            if (!continuationToken.empty()) {
                request.SetContinuationToken(continuationToken);
            }
            Aws::S3::Model::ListObjectsV2Outcome outcome =
s3Client.ListObjectsV2(
                    request);

            if (outcome.IsSuccess()) {
                const std::vector<Aws::S3::Model::Object> &objects =
                        outcome.GetResult().GetContents();
                allObjects.insert(allObjects.end(), objects.begin(),
objects.end());
                continuationToken =
outcome.GetResult().GetNextContinuationToken();
            }
            else {
```

```
                std::cerr << "Error listing objects. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                break;
            }
        } while (!continuationToken.empty());

        std::cout << "Data from your job is in " << allObjects.size() <<
                  " files in the S3 bucket, " << bucketName << "." << std::endl;

        for (size_t i = 0; i < allObjects.size(); ++i) {
            std::cout << "    " << i + 1 << ". " << allObjects[i].GetKey()
                      << std::endl;
        }

        int objectIndex = askQuestionForIntRange(
                std::string(
                        "Enter the number of a block to download it and see the
 first ") +
                std::to_string(LINES_OF_RUN_FILE_TO_DISPLAY) +
                " lines of JSON output in the block: ", 1,
                static_cast<int>(allObjects.size()));

        Aws::String objectKey = allObjects[objectIndex - 1].GetKey();

        std::stringstream stringStream;
        if (getObjectFromBucket(bucketName, objectKey, stringStream,
                                clientConfig)) {
            for (int i = 0; i < LINES_OF_RUN_FILE_TO_DISPLAY && stringStream; +
+i) {
                std::string line;
                std::getline(stringStream, line);
                std::cout << "    " << line << std::endl;
            }
        }
        else {
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME,
 bucketName,
                         clientConfig);
            return false;
        }
    }

    // 10. List all the jobs.
```

```
    Aws::String jobName;
    {
        Aws::Glue::Model::ListJobsRequest listJobsRequest;

        Aws::String nextToken;
        std::vector<Aws::String> allJobNames;

        do {
            if (!nextToken.empty()) {
                listJobsRequest.SetNextToken(nextToken);
            }
            Aws::Glue::Model::ListJobsOutcome listRunsOutcome = client.ListJobs(
                    listJobsRequest);

            if (listRunsOutcome.IsSuccess()) {
                const std::vector<Aws::String> &jobNames =
listRunsOutcome.GetResult().GetJobNames();
                allJobNames.insert(allJobNames.end(), jobNames.begin(),
jobNames.end());
                nextToken = listRunsOutcome.GetResult().GetNextToken();
            }
            else {
                std::cerr << "Error listing jobs. "
                          << listRunsOutcome.GetError().GetMessage()
                          << std::endl;
            }
        } while (!nextToken.empty());
        std::cout << "Your account has " << allJobNames.size() << " jobs."
                  << std::endl;
        for (size_t i = 0; i < allJobNames.size(); ++i) {
            std::cout << "   " << i + 1 << ". " << allJobNames[i] << std::endl;
        }
        int jobIndex = askQuestionForIntRange(
                Aws::String("Enter a number between 1 and ") +
                std::to_string(allJobNames.size()) +
                " to see the list of runs for a job: ",
                1, static_cast<int>(allJobNames.size()));

        jobName = allJobNames[jobIndex - 1];
    }

    // 11. Get the job runs for a job.
    Aws::String jobRunID;
    if (!jobName.empty()) {
```

```
        Aws::Glue::Model::GetJobRunsRequest getJobRunsRequest;
        getJobRunsRequest.SetJobName(jobName);

        Aws::String nextToken; // Used for pagination.
        std::vector<Aws::Glue::Model::JobRun> allJobRuns;
        do {
            if (!nextToken.empty()) {
                getJobRunsRequest.SetNextToken(nextToken);
            }
            Aws::Glue::Model::GetJobRunsOutcome jobRunsOutcome =
client.GetJobRuns(
                    getJobRunsRequest);

            if (jobRunsOutcome.IsSuccess()) {
                const std::vector<Aws::Glue::Model::JobRun> &jobRuns =
jobRunsOutcome.GetResult().GetJobRuns();
                allJobRuns.insert(allJobRuns.end(), jobRuns.begin(),
jobRuns.end());

                nextToken = jobRunsOutcome.GetResult().GetNextToken();
            }
            else {
                std::cerr << "Error getting job runs. "
                          << jobRunsOutcome.GetError().GetMessage()
                          << std::endl;
                break;
            }
        } while (!nextToken.empty());

        std::cout << "There are " << allJobRuns.size() << " runs in the job '"
                  <<
                  jobName << "'." << std::endl;

        for (size_t i = 0; i < allJobRuns.size(); ++i) {
            std::cout << "   " << i + 1 << ". " << allJobRuns[i].GetJobName()
                      << std::endl;
        }

        int runIndex = askQuestionForIntRange(
                Aws::String("Enter a number between 1 and ") +
                std::to_string(allJobRuns.size()) +
                " to see details for a run: ",
                1, static_cast<int>(allJobRuns.size()));
        jobRunID = allJobRuns[runIndex - 1].GetId();
```

```
    }

    // 12. Get a single job run.
    if (!jobRunID.empty()) {
        Aws::Glue::Model::GetJobRunRequest jobRunRequest;
        jobRunRequest.SetJobName(jobName);
        jobRunRequest.SetRunId(jobRunID);

        Aws::Glue::Model::GetJobRunOutcome jobRunOutcome = client.GetJobRun(
                jobRunRequest);

        if (jobRunOutcome.IsSuccess()) {
            std::cout << "Displaying the job run JSON description." << std::endl;
            std::cout
                    <<
 jobRunOutcome.GetResult().GetJobRun().Jsonize().View().WriteReadable()
                    << std::endl;
        }
        else {
            std::cerr << "Error get a job run. "
                      << jobRunOutcome.GetError().GetMessage()
                      << std::endl;
        }
    }

    return deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME,
 bucketName,
                        clientConfig);
}

//! Cleanup routine to delete created assets.
/*!
 \\sa deleteAssets()
 \param crawler: Name of an AWS Glue crawler.
 \param database: The name of an AWS Glue database.
 \param job: The name of an AWS Glue job.
 \param bucketName: The name of an S3 bucket.
 \param clientConfig: AWS client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::Glue::deleteAssets(const Aws::String &crawler, const Aws::String
 &database,
                                const Aws::String &job, const Aws::String
 &bucketName,
```

```
                                const Aws::Client::ClientConfiguration
&clientConfig) {
   const Aws::Glue::GlueClient client(clientConfig);
   bool result = true;

   // 13. Delete a job.
   if (!job.empty()) {
       Aws::Glue::Model::DeleteJobRequest request;
       request.SetJobName(job);

       Aws::Glue::Model::DeleteJobOutcome outcome = client.DeleteJob(request);


       if (outcome.IsSuccess()) {
           std::cout << "Successfully deleted the job." << std::endl;
       }
       else {
           std::cerr << "Error deleting the job. " <<
outcome.GetError().GetMessage()
                     << std::endl;
           result = false;
       }
   }

   // 14. Delete a database.
   if (!database.empty()) {
       Aws::Glue::Model::DeleteDatabaseRequest request;
       request.SetName(database);

       Aws::Glue::Model::DeleteDatabaseOutcome outcome = client.DeleteDatabase(
               request);

       if (outcome.IsSuccess()) {
           std::cout << "Successfully deleted the database." << std::endl;
       }
       else {
           std::cerr << "Error deleting database. " <<
outcome.GetError().GetMessage()
                     << std::endl;
           result = false;
       }
   }

   // 15. Delete a crawler.
```

```
    if (!crawler.empty()) {
        Aws::Glue::Model::DeleteCrawlerRequest request;
        request.SetName(crawler);

        Aws::Glue::Model::DeleteCrawlerOutcome outcome =
 client.DeleteCrawler(request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully deleted the crawler." << std::endl;
        }
        else {
            std::cerr << "Error deleting the crawler. "
                      << outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
    }

    // 16. Delete the job script and run data from the S3 bucket.
    result &= AwsDoc::Glue::deleteAllObjectsInS3Bucket(bucketName,
                                                       clientConfig);
    return result;
}

//! Routine which uploads a file to an S3 bucket.
/*!
 \\sa uploadFile()
 \param bucketName: An S3 bucket created in the setup.
 \param filePath: The path of the file to upload.
 \param fileName The name for the uploaded file.
 \param clientConfig: AWS client configuration.
 \return bool: Successful completion.
 */
bool
AwsDoc::Glue::uploadFile(const Aws::String &bucketName,
                         const Aws::String &filePath,
                         const Aws::String &fileName,
                         const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3_client(clientConfig);

    Aws::S3::Model::PutObjectRequest request;
    request.SetBucket(bucketName);
    request.SetKey(fileName);

    std::shared_ptr<Aws::IOStream> inputData =
```

```
            Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                                          filePath.c_str(),
                                          std::ios_base::in |
 std::ios_base::binary);

    if (!*inputData) {
        std::cerr << "Error unable to read file " << filePath << std::endl;
        return false;
    }

    request.SetBody(inputData);

    Aws::S3::Model::PutObjectOutcome outcome =
            s3_client.PutObject(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: PutObject: " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Added object '" << filePath << "' to bucket '"
                  << bucketName << "'." << std::endl;
    }

    return outcome.IsSuccess();
}

//! Routine which deletes all objects in an S3 bucket.
/*!
 \\sa deleteAllObjectsInS3Bucket()
 \param bucketName: The S3 bucket name.
 \param clientConfig: AWS client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::Glue::deleteAllObjectsInS3Bucket(const Aws::String &bucketName,
                                              const
 Aws::Client::ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::ListObjectsV2Request listObjectsRequest;
    listObjectsRequest.SetBucket(bucketName);

    Aws::String continuationToken; // Used for pagination.
    bool result = true;
    do {
```

```
        if (!continuationToken.empty()) {
            listObjectsRequest.SetContinuationToken(continuationToken);
        }

        Aws::S3::Model::ListObjectsV2Outcome listObjectsOutcome =
client.ListObjectsV2(
                listObjectsRequest);

        if (listObjectsOutcome.IsSuccess()) {
            const std::vector<Aws::S3::Model::Object> &objects =
listObjectsOutcome.GetResult().GetContents();
            if (!objects.empty()) {
                Aws::S3::Model::DeleteObjectsRequest deleteObjectsRequest;
                deleteObjectsRequest.SetBucket(bucketName);

                std::vector<Aws::S3::Model::ObjectIdentifier> objectIdentifiers;
                for (const Aws::S3::Model::Object &object: objects) {
                    objectIdentifiers.push_back(
                            Aws::S3::Model::ObjectIdentifier().WithKey(
                                    object.GetKey()));
                }
                Aws::S3::Model::Delete objectsDelete;
                objectsDelete.SetObjects(objectIdentifiers);
                objectsDelete.SetQuiet(true);
                deleteObjectsRequest.SetDelete(objectsDelete);

                Aws::S3::Model::DeleteObjectsOutcome deleteObjectsOutcome =
                        client.DeleteObjects(deleteObjectsRequest);

                if (!deleteObjectsOutcome.IsSuccess()) {
                    std::cerr << "Error deleting objects. " <<
                              deleteObjectsOutcome.GetError().GetMessage() <<
std::endl;
                    result = false;
                    break;
                }
                else {
                    std::cout << "Successfully deleted the objects." <<
std::endl;

                }
            }
            else {
                std::cout << "No objects to delete in '" << bucketName << "'."
```

```
                          << std::endl;
            }

            continuationToken =
 listObjectsOutcome.GetResult().GetNextContinuationToken();
        }
        else {
            std::cerr << "Error listing objects. "
                      << listObjectsOutcome.GetError().GetMessage() << std::endl;
            result = false;
            break;
        }
    } while (!continuationToken.empty());

    return result;
}

//! Routine which retrieves an object from an S3 bucket.
/*!
 \\sa getObjectFromBucket()
 \param bucketName: The S3 bucket name.
 \param objectKey: The object's name.
 \param objectStream: A stream to receive the retrieved data.
 \param clientConfig: AWS client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::Glue::getObjectFromBucket(const Aws::String &bucketName,
                                       const Aws::String &objectKey,
                                       std::ostream &objectStream,
                                       const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::GetObjectRequest request;
    request.SetBucket(bucketName);
    request.SetKey(objectKey);

    Aws::S3::Model::GetObjectOutcome outcome = client.GetObject(request);


    if (outcome.IsSuccess()) {
        std::cout << "Successfully retrieved '" << objectKey << "'." <<
 std::endl;
        auto &body = outcome.GetResult().GetBody();
        objectStream << body.rdbuf();
```

```
    }
    else {
        std::cerr << "Error retrieving object. " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Para obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK for C++ .
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun
  - GetJobRuns
  - GetTables
  - ListJobs
  - StartCrawler
  - StartJobRun

Java

SDK para Java 2.x

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
/**
 *
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * To set up the resources, see this documentation topic:
 *
 * https://docs.aws.amazon.com/glue/latest/ug/tutorial-add-crawler.html
 *
 * This example performs the following tasks:
 *
 * 1. Create a database.
 * 2. Create a crawler.
 * 3. Get a crawler.
 * 4. Start a crawler.
 * 5. Get a database.
 * 6. Get tables.
 * 7. Create a job.
 * 8. Start a job run.
 * 9. List all jobs.
 * 10. Get job runs.
 * 11. Delete a job.
 * 12. Delete a database.
 * 13. Delete a crawler.
 */

public class GlueScenario {
```

```
   public static final String DASHES = new String(new char[80]).replace("\0",
"-");

   public static void main(String[] args) throws InterruptedException {
       final String usage = """

               Usage:
                   <iam> <s3Path> <cron> <dbName> <crawlerName> <jobName>\s

               Where:
                   iam - The ARN of the IAM role that has AWS Glue and S3
permissions.\s
                   s3Path - The Amazon Simple Storage Service (Amazon S3) target
that contains data (for example, CSV data).
                   cron - A cron expression used to specify the schedule  (i.e.,
cron(15 12 * * ? *).
                   dbName - The database name.\s
                   crawlerName - The name of the crawler.\s
                   jobName - The name you assign to this job definition.
                   scriptLocation - The Amazon S3 path to a script that runs a
job.
                   locationUri - The location of the database
                   bucketNameSc - The Amazon S3 bucket name used when creating a
job
                   """;

       if (args.length != 9) {
           System.out.println(usage);
           System.exit(1);
       }

       String iam = args[0];
       String s3Path = args[1];
       String cron = args[2];
       String dbName = args[3];
       String crawlerName = args[4];
       String jobName = args[5];
       String scriptLocation = args[6];
       String locationUri = args[7];
       String bucketNameSc = args[8];

       Region region = Region.US_EAST_1;
       GlueClient glueClient = GlueClient.builder()
               .region(region)
```

```
                .build();
        System.out.println(DASHES);
        System.out.println("Welcome to the AWS Glue scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Create a database.");
        createDatabase(glueClient, dbName, locationUri);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Create a crawler.");
        createGlueCrawler(glueClient, iam, s3Path, cron, dbName, crawlerName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Get a crawler.");
        getSpecificCrawler(glueClient, crawlerName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Start a crawler.");
        startSpecificCrawler(glueClient, crawlerName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Get a database.");
        getSpecificDatabase(glueClient, dbName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("*** Wait 5 min for the tables to become available");
        TimeUnit.MINUTES.sleep(5);
        System.out.println("6. Get tables.");
        String myTableName = getGlueTables(glueClient, dbName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Create a job.");
        createJob(glueClient, jobName, iam, scriptLocation);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Start a Job run.");
```

```
        startJob(glueClient, jobName, dbName, myTableName, bucketNameSc);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. List all jobs.");
        getAllJobs(glueClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Get job runs.");
        getJobRuns(glueClient, jobName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Delete a job.");
        deleteJob(glueClient, jobName);
        System.out.println("*** Wait 5 MIN for the " + crawlerName + " to stop");
        TimeUnit.MINUTES.sleep(5);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Delete a database.");
        deleteDatabase(glueClient, dbName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Delete a crawler.");
        deleteSpecificCrawler(glueClient, crawlerName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Successfully completed the AWS Glue Scenario");
        System.out.println(DASHES);
    }

    public static void createDatabase(GlueClient glueClient, String dbName,
String locationUri) {
        try {
            DatabaseInput input = DatabaseInput.builder()
                    .description("Built with the AWS SDK for Java V2")
                    .name(dbName)
                    .locationUri(locationUri)
                    .build();
```

```
            CreateDatabaseRequest request = CreateDatabaseRequest.builder()
                    .databaseInput(input)
                    .build();

            glueClient.createDatabase(request);
            System.out.println(dbName + " was successfully created");

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void createGlueCrawler(GlueClient glueClient,
            String iam,
            String s3Path,
            String cron,
            String dbName,
            String crawlerName) {

        try {
            S3Target s3Target = S3Target.builder()
                    .path(s3Path)
                    .build();

            List<S3Target> targetList = new ArrayList<>();
            targetList.add(s3Target);
            CrawlerTargets targets = CrawlerTargets.builder()
                    .s3Targets(targetList)
                    .build();

            CreateCrawlerRequest crawlerRequest = CreateCrawlerRequest.builder()
                    .databaseName(dbName)
                    .name(crawlerName)
                    .description("Created by the AWS Glue Java API")
                    .targets(targets)
                    .role(iam)
                    .schedule(cron)
                    .build();

            glueClient.createCrawler(crawlerRequest);
            System.out.println(crawlerName + " was successfully created");

        } catch (GlueException e) {
```

```
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void getSpecificCrawler(GlueClient glueClient, String
crawlerName) {
        try {
            GetCrawlerRequest crawlerRequest = GetCrawlerRequest.builder()
                    .name(crawlerName)
                    .build();

            boolean ready = false;
            while (!ready) {
                GetCrawlerResponse response =
glueClient.getCrawler(crawlerRequest);
                String status = response.crawler().stateAsString();
                if (status.compareTo("READY") == 0) {
                    ready = true;
                }
                Thread.sleep(3000);
            }

            System.out.println("The crawler is now ready");

        } catch (GlueException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void startSpecificCrawler(GlueClient glueClient, String
crawlerName) {
        try {
            StartCrawlerRequest crawlerRequest = StartCrawlerRequest.builder()
                    .name(crawlerName)
                    .build();

            glueClient.startCrawler(crawlerRequest);
            System.out.println(crawlerName + " was successfully started!");

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
```

```
        }
    }

    public static void getSpecificDatabase(GlueClient glueClient, String
databaseName) {
        try {
            GetDatabaseRequest databasesRequest = GetDatabaseRequest.builder()
                    .name(databaseName)
                    .build();

            GetDatabaseResponse response =
glueClient.getDatabase(databasesRequest);
            Instant createDate = response.database().createTime();

            // Convert the Instant to readable date.
            DateTimeFormatter formatter =
DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
                    .withLocale(Locale.US)
                    .withZone(ZoneId.systemDefault());

            formatter.format(createDate);
            System.out.println("The create date of the database is " +
createDate);

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static String getGlueTables(GlueClient glueClient, String dbName) {
        String myTableName = "";
        try {
            GetTablesRequest tableRequest = GetTablesRequest.builder()
                    .databaseName(dbName)
                    .build();

            GetTablesResponse response = glueClient.getTables(tableRequest);
            List<Table> tables = response.tableList();
            if (tables.isEmpty()) {
                System.out.println("No tables were returned");
            } else {
                for (Table table : tables) {
                    myTableName = table.name();
```

```
                    System.out.println("Table name is: " + myTableName);
                }
            }

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return myTableName;
    }

    public static void startJob(GlueClient glueClient, String jobName, String
inputDatabase, String inputTable,
            String outBucket) {
        try {
            Map<String, String> myMap = new HashMap<>();
            myMap.put("--input_database", inputDatabase);
            myMap.put("--input_table", inputTable);
            myMap.put("--output_bucket_url", outBucket);

            StartJobRunRequest runRequest = StartJobRunRequest.builder()
                    .workerType(WorkerType.G_1_X)
                    .numberOfWorkers(10)
                    .arguments(myMap)
                    .jobName(jobName)
                    .build();

            StartJobRunResponse response = glueClient.startJobRun(runRequest);
            System.out.println("The request Id of the job is " +
response.responseMetadata().requestId());

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void createJob(GlueClient glueClient, String jobName, String
iam, String scriptLocation) {
        try {
            JobCommand command = JobCommand.builder()
                    .pythonVersion("3")
                    .name("glueetl")
                    .scriptLocation(scriptLocation)
```

```
                    .build();

            CreateJobRequest jobRequest = CreateJobRequest.builder()
                    .description("A Job created by using the AWS SDK for Java
V2")
                    .glueVersion("2.0")
                    .workerType(WorkerType.G_1_X)
                    .numberOfWorkers(10)
                    .name(jobName)
                    .role(iam)
                    .command(command)
                    .build();

            glueClient.createJob(jobRequest);
            System.out.println(jobName + " was successfully created.");

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void getAllJobs(GlueClient glueClient) {
        try {
            GetJobsRequest jobsRequest = GetJobsRequest.builder()
                    .maxResults(10)
                    .build();

            GetJobsResponse jobsResponse = glueClient.getJobs(jobsRequest);
            List<Job> jobs = jobsResponse.jobs();
            for (Job job : jobs) {
                System.out.println("Job name is : " + job.name());
                System.out.println("The job worker type is : " +
job.workerType().name());
            }

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void getJobRuns(GlueClient glueClient, String jobName) {
        try {
```

```
            GetJobRunsRequest runsRequest = GetJobRunsRequest.builder()
                    .jobName(jobName)
                    .maxResults(20)
                    .build();

            boolean jobDone = false;
            while (!jobDone) {
                GetJobRunsResponse response = glueClient.getJobRuns(runsRequest);
                List<JobRun> jobRuns = response.jobRuns();
                for (JobRun jobRun : jobRuns) {
                    String jobState = jobRun.jobRunState().name();
                    if (jobState.compareTo("SUCCEEDED") == 0) {
                        System.out.println(jobName + " has succeeded");
                        jobDone = true;

                    } else if (jobState.compareTo("STOPPED") == 0) {
                        System.out.println("Job run has stopped");
                        jobDone = true;

                    } else if (jobState.compareTo("FAILED") == 0) {
                        System.out.println("Job run has failed");
                        jobDone = true;

                    } else if (jobState.compareTo("TIMEOUT") == 0) {
                        System.out.println("Job run has timed out");
                        jobDone = true;

                    } else {
                        System.out.println("*** Job run state is " +
jobRun.jobRunState().name());
                        System.out.println("Job run Id is " + jobRun.id());
                        System.out.println("The Glue version is " +
jobRun.glueVersion());
                    }
                    TimeUnit.SECONDS.sleep(5);
                }
            }

        } catch (GlueException | InterruptedException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

```
    public static void deleteJob(GlueClient glueClient, String jobName) {
        try {
            DeleteJobRequest jobRequest = DeleteJobRequest.builder()
                    .jobName(jobName)
                    .build();

            glueClient.deleteJob(jobRequest);
            System.out.println(jobName + " was successfully deleted");

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteDatabase(GlueClient glueClient, String databaseName)
{
        try {
            DeleteDatabaseRequest request = DeleteDatabaseRequest.builder()
                    .name(databaseName)
                    .build();

            glueClient.deleteDatabase(request);
            System.out.println(databaseName + " was successfully deleted");

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteSpecificCrawler(GlueClient glueClient, String
crawlerName) {
        try {
            DeleteCrawlerRequest deleteCrawlerRequest =
DeleteCrawlerRequest.builder()
                    .name(crawlerName)
                    .build();

            glueClient.deleteCrawler(deleteCrawlerRequest);
            System.out.println(crawlerName + " was deleted");

        } catch (GlueException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
    }
}
```

- Para obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK for Java 2.x .
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun
  - GetJobRuns
  - GetTables
  - ListJobs
  - StartCrawler
  - StartJobRun

JavaScript

SDK para JavaScript (v3)

> **Note**
>
> Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Cree y ejecute un rastreador que rastree un bucket público de Amazon Simple Storage Service (Amazon S3) y genere una base de metadatos que describa los datos con formato CSV que encuentra.

```
const createCrawler = (name, role, dbName, tablePrefix, s3TargetPath) => {
  const client = new GlueClient({});

  const command = new CreateCrawlerCommand({
    Name: name,
    Role: role,
    DatabaseName: dbName,
    TablePrefix: tablePrefix,
    Targets: {
      S3Targets: [{ Path: s3TargetPath }],
    },
  });

  return client.send(command);
};

const getCrawler = (name) => {
  const client = new GlueClient({});

  const command = new GetCrawlerCommand({
    Name: name,
  });

  return client.send(command);
};

const startCrawler = (name) => {
  const client = new GlueClient({});

  const command = new StartCrawlerCommand({
    Name: name,
  });

  return client.send(command);
};

const crawlerExists = async ({ getCrawler }, crawlerName) => {
  try {
    await getCrawler(crawlerName);
```

```
    return true;
  } catch {
    return false;
  }
};

/**
 * @param {{ createCrawler: import('../../../actions/create-
crawler.js').createCrawler}} actions
 */
const makeCreateCrawlerStep = (actions) => async (context) => {
  if (await crawlerExists(actions, process.env.CRAWLER_NAME)) {
    log("Crawler already exists. Skipping creation.");
  } else {
    await actions.createCrawler(
      process.env.CRAWLER_NAME,
      process.env.ROLE_NAME,
      process.env.DATABASE_NAME,
      process.env.TABLE_PREFIX,
      process.env.S3_TARGET_PATH,
    );

    log("Crawler created successfully.", { type: "success" });
  }

  return { ...context };
};

/**
 * @param {(name: string) => Promise<import('@aws-sdk/client-
glue').GetCrawlerCommandOutput>} getCrawler
 * @param {string} crawlerName
 */
const waitForCrawler = async (getCrawler, crawlerName) => {
  const waitTimeInSeconds = 30;
  const { Crawler } = await getCrawler(crawlerName);

  if (!Crawler) {
    throw new Error(`Crawler with name ${crawlerName} not found.`);
  }

  if (Crawler.State === "READY") {
    return;
  }
```

```
  log(`Crawler is ${Crawler.State}. Waiting ${waitTimeInSeconds} seconds...`);
  await wait(waitTimeInSeconds);
  return waitForCrawler(getCrawler, crawlerName);
};

const makeStartCrawlerStep =
  ({ startCrawler, getCrawler }) =>
  async (context) => {
    log("Starting crawler.");
    await startCrawler(process.env.CRAWLER_NAME);
    log("Crawler started.", { type: "success" });

    log("Waiting for crawler to finish running. This can take a while.");
    await waitForCrawler(getCrawler, process.env.CRAWLER_NAME);
    log("Crawler ready.", { type: "success" });

    return { ...context };
  };
```

Enumere información sobre bases de datos y tablas en su AWS Glue Data Catalog.

```
const getDatabase = (name) => {
  const client = new GlueClient({});

  const command = new GetDatabaseCommand({
    Name: name,
  });

  return client.send(command);
};

const getTables = (databaseName) => {
  const client = new GlueClient({});

  const command = new GetTablesCommand({
    DatabaseName: databaseName,
  });

  return client.send(command);
};
```

```
const makeGetDatabaseStep =
  ({ getDatabase }) =>
  async (context) => {
    const {
      Database: { Name },
    } = await getDatabase(process.env.DATABASE_NAME);
    log(`Database: ${Name}`);
    return { ...context };
  };

/**
 * @param {{ getTables: () => Promise<import('@aws-sdk/client-
glue').GetTablesCommandOutput}} config
 */
const makeGetTablesStep =
  ({ getTables }) =>
  async (context) => {
    const { TableList } = await getTables(process.env.DATABASE_NAME);
    log("Tables:");
    log(TableList.map((table) => `  • ${table.Name}\n`));
    return { ...context };
  };
```

Cree y ejecute un trabajo que extraiga datos CSV del bucket de Amazon S3 de origen, los transforme quitando campos y cambiándoles el nombre, y cargue el resultado con formato JSON en otro bucket de Amazon S3.

```
const createJob = (name, role, scriptBucketName, scriptKey) => {
  const client = new GlueClient({});

  const command = new CreateJobCommand({
    Name: name,
    Role: role,
    Command: {
      Name: "glueetl",
      PythonVersion: "3",
      ScriptLocation: `s3://${scriptBucketName}/${scriptKey}`,
    },
    GlueVersion: "3.0",
  });

  return client.send(command);
```

```
};

const startJobRun = (jobName, dbName, tableName, bucketName) => {
  const client = new GlueClient({});

  const command = new StartJobRunCommand({
    JobName: jobName,
    Arguments: {
      "--input_database": dbName,
      "--input_table": tableName,
      "--output_bucket_url": `s3://${bucketName}/`,
    },
  });

  return client.send(command);
};

const makeCreateJobStep =
  ({ createJob }) =>
  async (context) => {
    log("Creating Job.");
    await createJob(
      process.env.JOB_NAME,
      process.env.ROLE_NAME,
      process.env.BUCKET_NAME,
      process.env.PYTHON_SCRIPT_KEY,
    );
    log("Job created.", { type: "success" });

    return { ...context };
  };

/**
 * @param {(name: string, runId: string) => Promise<import('@aws-sdk/client-
glue').GetJobRunCommandOutput> }  getJobRun
 * @param {string} jobName
 * @param {string} jobRunId
 */
const waitForJobRun = async (getJobRun, jobName, jobRunId) => {
  const waitTimeInSeconds = 30;
  const { JobRun } = await getJobRun(jobName, jobRunId);

  if (!JobRun) {
    throw new Error(`Job run with id ${jobRunId} not found.`);
```

```
  }

  switch (JobRun.JobRunState) {
    case "FAILED":
    case "TIMEOUT":
    case "STOPPED":
      throw new Error(
        `Job ${JobRun.JobRunState}. Error: ${JobRun.ErrorMessage}`,
      );
    case "RUNNING":
      break;
    case "SUCCEEDED":
      return;
    default:
      throw new Error(`Unknown job run state: ${JobRun.JobRunState}`);
  }

  log(
    `Job ${JobRun.JobRunState}. Waiting ${waitTimeInSeconds} more seconds...`,
  );
  await wait(waitTimeInSeconds);
  return waitForJobRun(getJobRun, jobName, jobRunId);
};

/**
 * @param {{ prompter: { prompt: () => Promise<{ shouldOpen: boolean }>} }}
 context
 */
const promptToOpen = async (context) => {
  const { shouldOpen } = await context.prompter.prompt({
    name: "shouldOpen",
    type: "confirm",
    message: "Open the output bucket in your browser?",
  });

  if (shouldOpen) {
    return open(
      `https://s3.console.aws.amazon.com/s3/buckets/${process.env.BUCKET_NAME} to
 view the output.`,
    );
  }
};

const makeStartJobRunStep =
```

```
  ({ startJobRun, getJobRun }) =>
  async (context) => {
    log("Starting job.");
    const { JobRunId } = await startJobRun(
      process.env.JOB_NAME,
      process.env.DATABASE_NAME,
      process.env.TABLE_NAME,
      process.env.BUCKET_NAME,
    );
    log("Job started.", { type: "success" });

    log("Waiting for job to finish running. This can take a while.");
    await waitForJobRun(getJobRun, process.env.JOB_NAME, JobRunId);
    log("Job run succeeded.", { type: "success" });

    await promptToOpen(context);

    return { ...context };
  };
```

Incluya información sobre las ejecuciones de trabajos y vea algunos de los datos transformados.

```
const getJobRuns = (jobName) => {
  const client = new GlueClient({});
  const command = new GetJobRunsCommand({
    JobName: jobName,
  });

  return client.send(command);
};

const getJobRun = (jobName, jobRunId) => {
  const client = new GlueClient({});
  const command = new GetJobRunCommand({
    JobName: jobName,
    RunId: jobRunId,
  });

  return client.send(command);
};
```

```
/**
 * @typedef {{ prompter: { prompt: () => Promise<{jobName: string}> } }} Context
 */

/**
 * @typedef {() => Promise<import('@aws-sdk/client-
glue').GetJobRunCommandOutput>} getJobRun
 */

/**
 * @typedef {() => Promise<import('@aws-sdk/client-
glue').GetJobRunsCommandOutput} getJobRuns
 */

/**
 *
 * @param {getJobRun} getJobRun
 * @param {string} jobName
 * @param {string} jobRunId
 */
const logJobRunDetails = async (getJobRun, jobName, jobRunId) => {
  const { JobRun } = await getJobRun(jobName, jobRunId);
  log(JobRun, { type: "object" });
};

/**
 *
 * @param {{getJobRuns: getJobRuns, getJobRun: getJobRun }} funcs
 */
const makePickJobRunStep =
  ({ getJobRuns, getJobRun }) =>
  async (/** @type { Context } */ context) => {
    if (context.selectedJobName) {
      const { JobRuns } = await getJobRuns(context.selectedJobName);

      const { jobRunId } = await context.prompter.prompt({
        name: "jobRunId",
        type: "list",
        message: "Select a job run to see details.",
        choices: JobRuns.map((run) => run.Id),
      });

      logJobRunDetails(getJobRun, context.selectedJobName, jobRunId);
    }
```

```
    return { ...context };
  };
```

Elimine todos los recursos creados en la demostración.

```
const deleteJob = (jobName) => {
  const client = new GlueClient({});

  const command = new DeleteJobCommand({
    JobName: jobName,
  });

  return client.send(command);
};

const deleteTable = (databaseName, tableName) => {
  const client = new GlueClient({});

  const command = new DeleteTableCommand({
    DatabaseName: databaseName,
    Name: tableName,
  });

  return client.send(command);
};

const deleteDatabase = (databaseName) => {
  const client = new GlueClient({});

  const command = new DeleteDatabaseCommand({
    Name: databaseName,
  });

  return client.send(command);
};

const deleteCrawler = (crawlerName) => {
  const client = new GlueClient({});

  const command = new DeleteCrawlerCommand({
    Name: crawlerName,
```

```
  });

  return client.send(command);
};

/**
 *
 * @param {import('../../../actions/delete-job.js').deleteJob} deleteJobFn
 * @param {string[]} jobNames
 * @param {{ prompter: { prompt: () => Promise<any> }}} context
 */
const handleDeleteJobs = async (deleteJobFn, jobNames, context) => {
  /**
   * @type {{ selectedJobNames: string[] }}
   */
  const { selectedJobNames } = await context.prompter.prompt({
    name: "selectedJobNames",
    type: "checkbox",
    message: "Let's clean up jobs. Select jobs to delete.",
    choices: jobNames,
  });

  if (selectedJobNames.length === 0) {
    log("No jobs selected.");
  } else {
    log("Deleting jobs.");
    await Promise.all(
      selectedJobNames.map((n) => deleteJobFn(n).catch(console.error)),
    );
    log("Jobs deleted.", { type: "success" });
  }
};

/**
 * @param {{
 *   listJobs: import('../../../actions/list-jobs.js').listJobs,
 *   deleteJob: import('../../../actions/delete-job.js').deleteJob
 * }} config
 */
const makeCleanUpJobsStep =
  ({ listJobs, deleteJob }) =>
  async (context) => {
    const { JobNames } = await listJobs();
    if (JobNames.length > 0) {
```

```
      await handleDeleteJobs(deleteJob, JobNames, context);
    }

    return { ...context };
  };

/**
 * @param {import('../../../actions/delete-table.js').deleteTable} deleteTable
 * @param {string} databaseName
 * @param {string[]} tableNames
 */
const deleteTables = (deleteTable, databaseName, tableNames) =>
  Promise.all(
    tableNames.map((tableName) =>
      deleteTable(databaseName, tableName).catch(console.error),
    ),
  );

/**
 * @param {{
 *  getTables: import('../../../actions/get-tables.js').getTables,
 *  deleteTable: import('../../../actions/delete-table.js').deleteTable
 * }} config
 */
const makeCleanUpTablesStep =
  ({ getTables, deleteTable }) =>
  /**
   * @param {{ prompter: { prompt: () => Promise<any>}}} context
   */
  async (context) => {
    const { TableList } = await getTables(process.env.DATABASE_NAME).catch(
      () => ({ TableList: null }),
    );

    if (TableList && TableList.length > 0) {
      /**
       * @type {{ tableNames: string[] }}
       */
      const { tableNames } = await context.prompter.prompt({
        name: "tableNames",
        type: "checkbox",
        message: "Let's clean up tables. Select tables to delete.",
        choices: TableList.map((t) => t.Name),
      });
```

```
      if (tableNames.length === 0) {
        log("No tables selected.");
      } else {
        log("Deleting tables.");
        await deleteTables(deleteTable, process.env.DATABASE_NAME, tableNames);
        log("Tables deleted.", { type: "success" });
      }
    }

    return { ...context };
  };

/**
 * @param {import('../../../actions/delete-database.js').deleteDatabase}
 deleteDatabase
 * @param {string[]} databaseNames
 */
const deleteDatabases = (deleteDatabase, databaseNames) =>
  Promise.all(
    databaseNames.map((dbName) => deleteDatabase(dbName).catch(console.error)),
  );

/**
 * @param {{
 *   getDatabases: import('../../../actions/get-databases.js').getDatabases
 *   deleteDatabase: import('../../../actions/delete-database.js').deleteDatabase
 * }} config
 */
const makeCleanUpDatabasesStep =
  ({ getDatabases, deleteDatabase }) =>
  /**
   * @param {{ prompter: { prompt: () => Promise<any>}} context
   */
  async (context) => {
    const { DatabaseList } = await getDatabases();

    if (DatabaseList.length > 0) {
      /** @type {{ dbNames: string[] }} */
      const { dbNames } = await context.prompter.prompt({
        name: "dbNames",
        type: "checkbox",
        message: "Let's clean up databases. Select databases to delete.",
        choices: DatabaseList.map((db) => db.Name),
```

```
      });

      if (dbNames.length === 0) {
        log("No databases selected.");
      } else {
        log("Deleting databases.");
        await deleteDatabases(deleteDatabase, dbNames);
        log("Databases deleted.", { type: "success" });
      }
    }

    return { ...context };
  };

const cleanUpCrawlerStep = async (context) => {
  log(`Deleting crawler.`);

  try {
    await deleteCrawler(process.env.CRAWLER_NAME);
    log("Crawler deleted.", { type: "success" });
  } catch (err) {
    if (err.name === "EntityNotFoundException") {
      log(`Crawler is already deleted.`);
    } else {
      throw err;
    }
  }

  return { ...context };
};
```

- Para obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK for JavaScript .
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable

- GetCrawler
- GetDatabase
- GetDatabases
- GetJob
- GetJobRun
- GetJobRuns
- GetTables
- ListJobs
- StartCrawler
- StartJobRun

Kotlin

SDK para Kotlin

> **Note**
>
> Hay más información sobre GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
suspend fun main(args: Array<String>) {
    val usage = """
        Usage:
            <iam> <s3Path> <cron> <dbName> <crawlerName> <jobName>
 <scriptLocation> <locationUri>

        Where:
            iam - The Amazon Resource Name (ARN) of the AWS Identity and Access
 Management (IAM) role that has AWS Glue and Amazon Simple Storage Service
 (Amazon S3) permissions.
            s3Path - The Amazon Simple Storage Service (Amazon S3) target that
 contains data (for example, CSV data).
            cron - A cron expression used to specify the schedule (for example,
 cron(15 12 * * ? *).
            dbName - The database name.
            crawlerName - The name of the crawler.
```

```
            jobName - The name you assign to this job definition.
            scriptLocation - Specifies the Amazon S3 path to a script that runs a
 job.
            locationUri - Specifies the location of the database
        """

    if (args.size != 8) {
        println(usage)
        exitProcess(1)
    }

    val iam = args[0]
    val s3Path = args[1]
    val cron = args[2]
    val dbName = args[3]
    val crawlerName = args[4]
    val jobName = args[5]
    val scriptLocation = args[6]
    val locationUri = args[7]

    println("About to start the AWS Glue Scenario")
    createDatabase(dbName, locationUri)
    createCrawler(iam, s3Path, cron, dbName, crawlerName)
    getCrawler(crawlerName)
    startCrawler(crawlerName)
    getDatabase(dbName)
    getGlueTables(dbName)
    createJob(jobName, iam, scriptLocation)
    startJob(jobName)
    getJobs()
    getJobRuns(jobName)
    deleteJob(jobName)
    println("*** Wait for 5 MIN so the $crawlerName is ready to be deleted")
    TimeUnit.MINUTES.sleep(5)
    deleteMyDatabase(dbName)
    deleteCrawler(crawlerName)
}

suspend fun createDatabase(
    dbName: String?,
    locationUriVal: String?,
) {
    val input =
        DatabaseInput {
```

```
            description = "Built with the AWS SDK for Kotlin"
            name = dbName
            locationUri = locationUriVal
        }

    val request =
        CreateDatabaseRequest {
            databaseInput = input
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.createDatabase(request)
        println("The database was successfully created")
    }
}

suspend fun createCrawler(
    iam: String?,
    s3Path: String?,
    cron: String?,
    dbName: String?,
    crawlerName: String,
) {
    val s3Target =
        S3Target {
            path = s3Path
        }

    val targetList = ArrayList<S3Target>()
    targetList.add(s3Target)

    val targetOb =
        CrawlerTargets {
            s3Targets = targetList
        }

    val crawlerRequest =
        CreateCrawlerRequest {
            databaseName = dbName
            name = crawlerName
            description = "Created by the AWS Glue Java API"
            targets = targetOb
            role = iam
            schedule = cron
```

```
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.createCrawler(crawlerRequest)
        println("$crawlerName was successfully created")
    }
}

suspend fun getCrawler(crawlerName: String?) {
    val request =
        GetCrawlerRequest {
            name = crawlerName
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getCrawler(request)
        val role = response.crawler?.role
        println("The role associated with this crawler is $role")
    }
}

suspend fun startCrawler(crawlerName: String) {
    val crawlerRequest =
        StartCrawlerRequest {
            name = crawlerName
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.startCrawler(crawlerRequest)
        println("$crawlerName was successfully started.")
    }
}

suspend fun getDatabase(databaseName: String?) {
    val request =
        GetDatabaseRequest {
            name = databaseName
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getDatabase(request)
        val dbDesc = response.database?.description
        println("The database description is $dbDesc")
    }
```

```
}

suspend fun getGlueTables(dbName: String?) {
    val tableRequest =
        GetTablesRequest {
            databaseName = dbName
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getTables(tableRequest)
        response.tableList?.forEach { tableName ->
            println("Table name is ${tableName.name}")
        }
    }
}

suspend fun startJob(jobNameVal: String?) {
    val runRequest =
        StartJobRunRequest {
            workerType = WorkerType.G1X
            numberOfWorkers = 10
            jobName = jobNameVal
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.startJobRun(runRequest)
        println("The job run Id is ${response.jobRunId}")
    }
}

suspend fun createJob(
    jobName: String,
    iam: String?,
    scriptLocationVal: String?,
) {
    val commandOb =
        JobCommand {
            pythonVersion = "3"
            name = "MyJob1"
            scriptLocation = scriptLocationVal
        }

    val jobRequest =
        CreateJobRequest {
```

```
            description = "A Job created by using the AWS SDK for Java V2"
            glueVersion = "2.0"
            workerType = WorkerType.G1X
            numberOfWorkers = 10
            name = jobName
            role = iam
            command = commandOb
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.createJob(jobRequest)
        println("$jobName was successfully created.")
    }
}

suspend fun getJobs() {
    val request =
        GetJobsRequest {
            maxResults = 10
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getJobs(request)
        response.jobs?.forEach { job ->
            println("Job name is ${job.name}")
        }
    }
}

suspend fun getJobRuns(jobNameVal: String?) {
    val request =
        GetJobRunsRequest {
            jobName = jobNameVal
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        val response = glueClient.getJobRuns(request)
        response.jobRuns?.forEach { job ->
            println("Job name is ${job.jobName}")
        }
    }
}

suspend fun deleteJob(jobNameVal: String) {
```

```
    val jobRequest =
        DeleteJobRequest {
            jobName = jobNameVal
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.deleteJob(jobRequest)
        println("$jobNameVal was successfully deleted")
    }
}

suspend fun deleteMyDatabase(databaseName: String) {
    val request =
        DeleteDatabaseRequest {
            name = databaseName
        }

    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.deleteDatabase(request)
        println("$databaseName was successfully deleted")
    }
}

suspend fun deleteCrawler(crawlerName: String) {
    val request =
        DeleteCrawlerRequest {
            name = crawlerName
        }
    GlueClient { region = "us-east-1" }.use { glueClient ->
        glueClient.deleteCrawler(request)
        println("$crawlerName was deleted")
    }
}
```

- Para obtener información acerca de la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para Kotlin.
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase

- DeleteJob
- DeleteTable
- GetCrawler
- GetDatabase
- GetDatabases
- GetJob
- GetJobRun
- GetJobRuns
- GetTables
- ListJobs
- StartCrawler
- StartJobRun

PHP

SDK para PHP

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

```
namespace Glue;

use Aws\Glue\GlueClient;
use Aws\S3\S3Client;
use AwsUtilities\AWSServiceClass;
use GuzzleHttp\Psr7\Stream;
use Iam\IAMService;

class GettingStartedWithGlue
{
    public function run()
    {
        echo("\n");
        echo("--------------------------------------\n");
```

```
        print("Welcome to the AWS Glue getting started demo using PHP!\n");
        echo("-------------------------------------\n");

        $clientArgs = [
            'region' => 'us-west-2',
            'version' => 'latest',
            'profile' => 'default',
        ];
        $uniqid = uniqid();

        $glueClient = new GlueClient($clientArgs);
        $glueService = new GlueService($glueClient);
        $iamService = new IAMService();
        $crawlerName = "example-crawler-test-" . $uniqid;

        AWSServiceClass::$waitTime = 5;
        AWSServiceClass::$maxWaitAttempts = 20;

        $role = $iamService->getRole("AWSGlueServiceRole-DocExample");

        $databaseName = "doc-example-database-$uniqid";
        $path = 's3://crawler-public-us-east-1/flight/2016/csv';
        $glueService->createCrawler($crawlerName, $role['Role']['Arn'],
$databaseName, $path);
        $glueService->startCrawler($crawlerName);

        echo "Waiting for crawler";
        do {
            $crawler = $glueService->getCrawler($crawlerName);
            echo ".";
            sleep(10);
        } while ($crawler['Crawler']['State'] != "READY");
        echo "\n";

        $database = $glueService->getDatabase($databaseName);
        echo "Found a database named " . $database['Database']['Name'] . "\n";

        //Upload job script
        $s3client = new S3Client($clientArgs);
        $bucketName = "test-glue-bucket-" . $uniqid;
        $s3client->createBucket([
            'Bucket' => $bucketName,
            'CreateBucketConfiguration' => ['LocationConstraint' => 'us-west-2'],
        ]);
```

```
        $s3client->putObject([
            'Bucket' => $bucketName,
            'Key' => 'run_job.py',
            'SourceFile' => __DIR__ . '/flight_etl_job_script.py'
        ]);
        $s3client->putObject([
            'Bucket' => $bucketName,
            'Key' => 'setup_scenario_getting_started.yaml',
            'SourceFile' => __DIR__ . '/setup_scenario_getting_started.yaml'
        ]);

        $tables = $glueService->getTables($databaseName);

        $jobName = 'test-job-' . $uniqid;
        $scriptLocation = "s3://$bucketName/run_job.py";
        $job = $glueService->createJob($jobName, $role['Role']['Arn'],
$scriptLocation);

        $outputBucketUrl = "s3://$bucketName";
        $runId = $glueService->startJobRun($jobName, $databaseName, $tables,
$outputBucketUrl)['JobRunId'];

        echo "waiting for job";
        do {
            $jobRun = $glueService->getJobRun($jobName, $runId);
            echo ".";
            sleep(10);
        } while (!array_intersect([$jobRun['JobRun']['JobRunState']],
['SUCCEEDED', 'STOPPED', 'FAILED', 'TIMEOUT']));
        echo "\n";

        $jobRuns = $glueService->getJobRuns($jobName);

        $objects = $s3client->listObjects([
            'Bucket' => $bucketName,
        ])['Contents'];

        foreach ($objects as $object) {
            echo $object['Key'] . "\n";
        }

        echo "Downloading " . $objects[1]['Key'] . "\n";
        /** @var Stream $downloadObject */
```

```
        $downloadObject = $s3client->getObject([
            'Bucket' => $bucketName,
            'Key' => $objects[1]['Key'],
        ])['Body']->getContents();
        echo "Here is the first 1000 characters in the object.";
        echo substr($downloadObject, 0, 1000);

        $jobs = $glueService->listJobs();
        echo "Current jobs:\n";
        foreach ($jobs['JobNames'] as $jobsName) {
            echo "{$jobsName}\n";
        }

        echo "Delete the job.\n";
        $glueClient->deleteJob([
            'JobName' => $job['Name'],
        ]);

        echo "Delete the tables.\n";
        foreach ($tables['TableList'] as $table) {
            $glueService->deleteTable($table['Name'], $databaseName);
        }

        echo "Delete the databases.\n";
        $glueClient->deleteDatabase([
            'Name' => $databaseName,
        ]);

        echo "Delete the crawler.\n";
        $glueClient->deleteCrawler([
            'Name' => $crawlerName,
        ]);

        $deleteObjects = $s3client->listObjectsV2([
            'Bucket' => $bucketName,
        ]);
        echo "Delete all objects in the bucket.\n";
        $deleteObjects = $s3client->deleteObjects([
            'Bucket' => $bucketName,
            'Delete' => [
                'Objects' => $deleteObjects['Contents'],
            ]
        ]);
        echo "Delete the bucket.\n";
```

```
        $s3client->deleteBucket(['Bucket' => $bucketName]);

        echo "This job was brought to you by the number $uniqid\n";
    }
}

namespace Glue;

use Aws\Glue\GlueClient;
use Aws\Result;

use function PHPUnit\Framework\isEmpty;

class GlueService extends \AwsUtilities\AWSServiceClass
{
    protected GlueClient $glueClient;

    public function __construct($glueClient)
    {
        $this->glueClient = $glueClient;
    }

    public function getCrawler($crawlerName)
    {
        return $this->customWaiter(function () use ($crawlerName) {
            return $this->glueClient->getCrawler([
                'Name' => $crawlerName,
            ]);
        });
    }

    public function createCrawler($crawlerName, $role, $databaseName, $path):
 Result
    {
        return $this->customWaiter(function () use ($crawlerName, $role,
 $databaseName, $path) {
            return $this->glueClient->createCrawler([
                'Name' => $crawlerName,
                'Role' => $role,
                'DatabaseName' => $databaseName,
                'Targets' => [
                    'S3Targets' =>
                        [[
                            'Path' => $path,
```

```
                        ]]
                ],
            ]);
        });
    }

    public function startCrawler($crawlerName): Result
    {
        return $this->glueClient->startCrawler([
            'Name' => $crawlerName,
        ]);
    }

    public function getDatabase(string $databaseName): Result
    {
        return $this->customWaiter(function () use ($databaseName) {
            return $this->glueClient->getDatabase([
                'Name' => $databaseName,
            ]);
        });
    }

    public function getTables($databaseName): Result
    {
        return $this->glueClient->getTables([
            'DatabaseName' => $databaseName,
        ]);
    }

    public function createJob($jobName, $role, $scriptLocation, $pythonVersion =
'3', $glueVersion = '3.0'): Result
    {
        return $this->glueClient->createJob([
            'Name' => $jobName,
            'Role' => $role,
            'Command' => [
                'Name' => 'glueetl',
                'ScriptLocation' => $scriptLocation,
                'PythonVersion' => $pythonVersion,
            ],
            'GlueVersion' => $glueVersion,
        ]);
    }
```

```
   public function startJobRun($jobName, $databaseName, $tables,
$outputBucketUrl): Result
   {
       return $this->glueClient->startJobRun([
           'JobName' => $jobName,
           'Arguments' => [
               'input_database' => $databaseName,
               'input_table' => $tables['TableList'][0]['Name'],
               'output_bucket_url' => $outputBucketUrl,
               '--input_database' => $databaseName,
               '--input_table' => $tables['TableList'][0]['Name'],
               '--output_bucket_url' => $outputBucketUrl,
           ],
       ]);
   }

   public function listJobs($maxResults = null, $nextToken = null, $tags = []):
Result
   {
       $arguments = [];
       if ($maxResults) {
           $arguments['MaxResults'] = $maxResults;
       }
       if ($nextToken) {
           $arguments['NextToken'] = $nextToken;
       }
       if (!empty($tags)) {
           $arguments['Tags'] = $tags;
       }
       return $this->glueClient->listJobs($arguments);
   }

   public function getJobRuns($jobName, $maxResults = 0, $nextToken = ''):
Result
   {
       $arguments = ['JobName' => $jobName];
       if ($maxResults) {
           $arguments['MaxResults'] = $maxResults;
       }
       if ($nextToken) {
           $arguments['NextToken'] = $nextToken;
       }
       return $this->glueClient->getJobRuns($arguments);
   }
```

```
    public function getJobRun($jobName, $runId, $predecessorsIncluded = false):
 Result
    {
        return $this->glueClient->getJobRun([
            'JobName' => $jobName,
            'RunId' => $runId,
            'PredecessorsIncluded' => $predecessorsIncluded,
        ]);
    }

    public function deleteJob($jobName)
    {
        return $this->glueClient->deleteJob([
            'JobName' => $jobName,
        ]);
    }

    public function deleteTable($tableName, $databaseName)
    {
        return $this->glueClient->deleteTable([
            'DatabaseName' => $databaseName,
            'Name' => $tableName,
        ]);
    }

    public function deleteDatabase($databaseName)
    {
        return $this->glueClient->deleteDatabase([
            'Name' => $databaseName,
        ]);
    }

    public function deleteCrawler($crawlerName)
    {
        return $this->glueClient->deleteCrawler([
            'Name' => $crawlerName,
        ]);
    }
}
```

- Para obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK for PHP .
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun
  - GetJobRuns
  - GetTables
  - ListJobs
  - StartCrawler
  - StartJobRun

Python

SDK para Python (Boto3)

> **Note**
>
> Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Cree una clase que agrupe AWS Glue las funciones utilizadas en el escenario.

```
class GlueWrapper:
    """Encapsulates AWS Glue actions."""

    def __init__(self, glue_client):
```

```
        """
        :param glue_client: A Boto3 Glue client.
        """
        self.glue_client = glue_client


    def get_crawler(self, name):
        """
        Gets information about a crawler.

        :param name: The name of the crawler to look up.
        :return: Data about the crawler.
        """
        crawler = None
        try:
            response = self.glue_client.get_crawler(Name=name)
            crawler = response["Crawler"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityNotFoundException":
                logger.info("Crawler %s doesn't exist.", name)
            else:
                logger.error(
                    "Couldn't get crawler %s. Here's why: %s: %s",
                    name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return crawler


    def create_crawler(self, name, role_arn, db_name, db_prefix, s3_target):
        """
        Creates a crawler that can crawl the specified target and populate a
        database in your AWS Glue Data Catalog with metadata that describes the
data
        in the target.

        :param name: The name of the crawler.
        :param role_arn: The Amazon Resource Name (ARN) of an AWS Identity and
Access
                         Management (IAM) role that grants permission to let AWS
Glue
                         access the resources it needs.
```

```
        :param db_name: The name to give the database that is created by the
crawler.
        :param db_prefix: The prefix to give any database tables that are created
by
                          the crawler.
        :param s3_target: The URL to an S3 bucket that contains data that is
                          the target of the crawler.
        """
        try:
            self.glue_client.create_crawler(
                Name=name,
                Role=role_arn,
                DatabaseName=db_name,
                TablePrefix=db_prefix,
                Targets={"S3Targets": [{"Path": s3_target}]},
            )
        except ClientError as err:
            logger.error(
                "Couldn't create crawler. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def start_crawler(self, name):
        """
        Starts a crawler. The crawler crawls its configured target and creates
        metadata that describes the data it finds in the target data source.

        :param name: The name of the crawler to start.
        """
        try:
            self.glue_client.start_crawler(Name=name)
        except ClientError as err:
            logger.error(
                "Couldn't start crawler %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

```
    def get_database(self, name):
        """
        Gets information about a database in your Data Catalog.

        :param name: The name of the database to look up.
        :return: Information about the database.
        """
        try:
            response = self.glue_client.get_database(Name=name)
        except ClientError as err:
            logger.error(
                "Couldn't get database %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["Database"]


    def get_tables(self, db_name):
        """
        Gets a list of tables in a Data Catalog database.

        :param db_name: The name of the database to query.
        :return: The list of tables in the database.
        """
        try:
            response = self.glue_client.get_tables(DatabaseName=db_name)
        except ClientError as err:
            logger.error(
                "Couldn't get tables %s. Here's why: %s: %s",
                db_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["TableList"]


    def create_job(self, name, description, role_arn, script_location):
        """
```

```
        Creates a job definition for an extract, transform, and load (ETL) job
that can
        be run by AWS Glue.

        :param name: The name of the job definition.
        :param description: The description of the job definition.
        :param role_arn: The ARN of an IAM role that grants AWS Glue the
permissions
                         it requires to run the job.
        :param script_location: The Amazon S3 URL of a Python ETL script that is
run as
                                part of the job. The script defines how the data
is
                                transformed.
        """
        try:
            self.glue_client.create_job(
                Name=name,
                Description=description,
                Role=role_arn,
                Command={
                    "Name": "glueetl",
                    "ScriptLocation": script_location,
                    "PythonVersion": "3",
                },
                GlueVersion="3.0",
            )
        except ClientError as err:
            logger.error(
                "Couldn't create job %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def start_job_run(self, name, input_database, input_table,
output_bucket_name):
        """
        Starts a job run. A job run extracts data from the source, transforms it,
        and loads it to the output bucket.

        :param name: The name of the job definition.
```

```
        :param input_database: The name of the metadata database that contains
tables
                              that describe the source data. This is typically
created
                              by a crawler.
        :param input_table: The name of the table in the metadata database that
                            describes the source data.
        :param output_bucket_name: The S3 bucket where the output is written.
        :return: The ID of the job run.
        """
        try:
            # The custom Arguments that are passed to this function are used by
the
            # Python ETL script to determine the location of input and output
data.
            response = self.glue_client.start_job_run(
                JobName=name,
                Arguments={
                    "--input_database": input_database,
                    "--input_table": input_table,
                    "--output_bucket_url": f"s3://{output_bucket_name}/",
                },
            )
        except ClientError as err:
            logger.error(
                "Couldn't start job run %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["JobRunId"]


    def list_jobs(self):
        """
        Lists the names of job definitions in your account.

        :return: The list of job definition names.
        """
        try:
            response = self.glue_client.list_jobs()
        except ClientError as err:
```

```
            logger.error(
                "Couldn't list jobs. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["JobNames"]


    def get_job_runs(self, job_name):
        """
        Gets information about runs that have been performed for a specific job
        definition.

        :param job_name: The name of the job definition to look up.
        :return: The list of job runs.
        """
        try:
            response = self.glue_client.get_job_runs(JobName=job_name)
        except ClientError as err:
            logger.error(
                "Couldn't get job runs for %s. Here's why: %s: %s",
                job_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["JobRuns"]


    def get_job_run(self, name, run_id):
        """
        Gets information about a single job run.

        :param name: The name of the job definition for the run.
        :param run_id: The ID of the run.
        :return: Information about the run.
        """
        try:
            response = self.glue_client.get_job_run(JobName=name, RunId=run_id)
        except ClientError as err:
            logger.error(
```

```
                "Couldn't get job run %s/%s. Here's why: %s: %s",
                name,
                run_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["JobRun"]


    def delete_job(self, job_name):
        """
        Deletes a job definition. This also deletes data about all runs that are
        associated with this job definition.

        :param job_name: The name of the job definition to delete.
        """
        try:
            self.glue_client.delete_job(JobName=job_name)
        except ClientError as err:
            logger.error(
                "Couldn't delete job %s. Here's why: %s: %s",
                job_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete_table(self, db_name, table_name):
        """
        Deletes a table from a metadata database.

        :param db_name: The name of the database that contains the table.
        :param table_name: The name of the table to delete.
        """
        try:
            self.glue_client.delete_table(DatabaseName=db_name, Name=table_name)
        except ClientError as err:
            logger.error(
                "Couldn't delete table %s. Here's why: %s: %s",
                table_name,
                err.response["Error"]["Code"],
```

```
                err.response["Error"]["Message"],
            )
            raise


    def delete_database(self, name):
        """
        Deletes a metadata database from your Data Catalog.

        :param name: The name of the database to delete.
        """
        try:
            self.glue_client.delete_database(Name=name)
        except ClientError as err:
            logger.error(
                "Couldn't delete database %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete_crawler(self, name):
        """
        Deletes a crawler.

        :param name: The name of the crawler to delete.
        """
        try:
            self.glue_client.delete_crawler(Name=name)
        except ClientError as err:
            logger.error(
                "Couldn't delete crawler %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

Crear una clase que ejecute el escenario.

```
class GlueCrawlerJobScenario:
    """
    Encapsulates a scenario that shows how to create an AWS Glue crawler and job
 and use
    them to transform data from CSV to JSON format.
    """

    def __init__(self, glue_client, glue_service_role, glue_bucket):
        """
        :param glue_client: A Boto3 AWS Glue client.
        :param glue_service_role: An AWS Identity and Access Management (IAM)
 role
                                  that AWS Glue can assume to gain access to the
                                  resources it requires.
        :param glue_bucket: An S3 bucket that can hold a job script and output
 data
                            from AWS Glue job runs.
        """
        self.glue_client = glue_client
        self.glue_service_role = glue_service_role
        self.glue_bucket = glue_bucket

    @staticmethod
    def wait(seconds, tick=12):
        """
        Waits for a specified number of seconds, while also displaying an
 animated
        spinner.

        :param seconds: The number of seconds to wait.
        :param tick: The number of frames per second used to animate the spinner.
        """
        progress = "|/-\\"
        waited = 0
        while waited < seconds:
            for frame in range(tick):
                sys.stdout.write(f"\r{progress[frame % len(progress)]}")
                sys.stdout.flush()
                time.sleep(1 / tick)
            waited += 1

    def upload_job_script(self, job_script):
```

```
        """
        Uploads a Python ETL script to an S3 bucket. The script is used by the
AWS Glue
        job to transform data.

        :param job_script: The relative path to the job script.
        """
        try:
            self.glue_bucket.upload_file(Filename=job_script, Key=job_script)
            print(f"Uploaded job script '{job_script}' to the example bucket.")
        except S3UploadFailedError as err:
            logger.error("Couldn't upload job script. Here's why: %s", err)
            raise

    def run(self, crawler_name, db_name, db_prefix, data_source, job_script,
job_name):
        """
        Runs the scenario. This is an interactive experience that runs at a
command
        prompt and asks you for input throughout.

        :param crawler_name: The name of the crawler used in the scenario. If the
                             crawler does not exist, it is created.
        :param db_name: The name to give the metadata database created by the
crawler.
        :param db_prefix: The prefix to give tables added to the database by the
                          crawler.
        :param data_source: The location of the data source that is targeted by
the
                            crawler and extracted during job runs.
        :param job_script: The job script that is used to transform data during
job
                           runs.
        :param job_name: The name to give the job definition that is created
during the
                         scenario.
        """
        wrapper = GlueWrapper(self.glue_client)
        print(f"Checking for crawler {crawler_name}.")
        crawler = wrapper.get_crawler(crawler_name)
        if crawler is None:
            print(f"Creating crawler {crawler_name}.")
            wrapper.create_crawler(
                crawler_name,
```

```
                self.glue_service_role.arn,
                db_name,
                db_prefix,
                data_source,
            )
            print(f"Created crawler {crawler_name}.")
            crawler = wrapper.get_crawler(crawler_name)
        pprint(crawler)
        print("-" * 88)

        print(
            f"When you run the crawler, it crawls data stored in {data_source}
and "
            f"creates a metadata database in the AWS Glue Data Catalog that
describes "
            f"the data in the data source."
        )
        print("In this example, the source data is in CSV format.")
        ready = False
        while not ready:
            ready = Question.ask_question(
                "Ready to start the crawler? (y/n) ", Question.is_yesno
            )
        wrapper.start_crawler(crawler_name)
        print("Let's wait for the crawler to run. This typically takes a few
minutes.")
        crawler_state = None
        while crawler_state != "READY":
            self.wait(10)
            crawler = wrapper.get_crawler(crawler_name)
            crawler_state = crawler["State"]
            print(f"Crawler is {crawler['State']}.")
        print("-" * 88)

        database = wrapper.get_database(db_name)
        print(f"The crawler created database {db_name}:")
        pprint(database)
        print(f"The database contains these tables:")
        tables = wrapper.get_tables(db_name)
        for index, table in enumerate(tables):
            print(f"\t{index + 1}. {table['Name']}")
        table_index = Question.ask_question(
            f"Enter the number of a table to see more detail: ",
            Question.is_int,
```

```
            Question.in_range(1, len(tables)),
        )
        pprint(tables[table_index - 1])
        print("-" * 88)

        print(f"Creating job definition {job_name}.")
        wrapper.create_job(
            job_name,
            "Getting started example job.",
            self.glue_service_role.arn,
            f"s3://{self.glue_bucket.name}/{job_script}",
        )
        print("Created job definition.")
        print(
            f"When you run the job, it extracts data from {data_source},
transforms it "
            f"by using the {job_script} script, and loads the output into "
            f"S3 bucket {self.glue_bucket.name}."
        )
        print(
            "In this example, the data is transformed from CSV to JSON, and only
a few "
            "fields are included in the output."
        )
        job_run_status = None
        if Question.ask_question(f"Ready to run? (y/n) ", Question.is_yesno):
            job_run_id = wrapper.start_job_run(
                job_name, db_name, tables[0]["Name"], self.glue_bucket.name
            )
            print(f"Job {job_name} started. Let's wait for it to run.")
            while job_run_status not in ["SUCCEEDED", "STOPPED", "FAILED",
"TIMEOUT"]:
                self.wait(10)
                job_run = wrapper.get_job_run(job_name, job_run_id)
                job_run_status = job_run["JobRunState"]
                print(f"Job {job_name}/{job_run_id} is {job_run_status}.")
        print("-" * 88)

        if job_run_status == "SUCCEEDED":
            print(
                f"Data from your job run is stored in your S3 bucket
'{self.glue_bucket.name}':"
            )
            try:
```

```
                keys = [
                    obj.key for obj in
self.glue_bucket.objects.filter(Prefix="run-")
                ]
                for index, key in enumerate(keys):
                    print(f"\t{index + 1}: {key}")
                lines = 4
                key_index = Question.ask_question(
                    f"Enter the number of a block to download it and see the
first {lines} "
                    f"lines of JSON output in the block: ",
                    Question.is_int,
                    Question.in_range(1, len(keys)),
                )
                job_data = io.BytesIO()
                self.glue_bucket.download_fileobj(keys[key_index - 1], job_data)
                job_data.seek(0)
                for _ in range(lines):
                    print(job_data.readline().decode("utf-8"))
            except ClientError as err:
                logger.error(
                    "Couldn't get job run data. Here's why: %s: %s",
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
            print("-" * 88)

        job_names = wrapper.list_jobs()
        if job_names:
            print(f"Your account has {len(job_names)} jobs defined:")
            for index, job_name in enumerate(job_names):
                print(f"\t{index + 1}. {job_name}")
            job_index = Question.ask_question(
                f"Enter a number between 1 and {len(job_names)} to see the list
of runs for "
                f"a job: ",
                Question.is_int,
                Question.in_range(1, len(job_names)),
            )
            job_runs = wrapper.get_job_runs(job_names[job_index - 1])
            if job_runs:
                print(f"Found {len(job_runs)} runs for job {job_names[job_index -
1]}:")
```

```
                for index, job_run in enumerate(job_runs):
                    print(
                        f"\t{index + 1}. {job_run['JobRunState']} on "
                        f"{job_run['CompletedOn']:%Y-%m-%d %H:%M:%S}"
                    )
                run_index = Question.ask_question(
                    f"Enter a number between 1 and {len(job_runs)} to see details
for a run: ",
                    Question.is_int,
                    Question.in_range(1, len(job_runs)),
                )
                pprint(job_runs[run_index - 1])
            else:
                print(f"No runs found for job {job_names[job_index - 1]}")
        else:
            print("Your account doesn't have any jobs defined.")
        print("-" * 88)

        print(
            f"Let's clean up. During this example we created job definition
'{job_name}'."
        )
        if Question.ask_question(
            "Do you want to delete the definition and all runs? (y/n) ",
            Question.is_yesno,
        ):
            wrapper.delete_job(job_name)
            print(f"Job definition '{job_name}' deleted.")
        tables = wrapper.get_tables(db_name)
        print(f"We also created database '{db_name}' that contains these
tables:")
        for table in tables:
            print(f"\t{table['Name']}")
        if Question.ask_question(
            "Do you want to delete the tables and the database? (y/n) ",
            Question.is_yesno,
        ):
            for table in tables:
                wrapper.delete_table(db_name, table["Name"])
                print(f"Deleted table {table['Name']}.")
            wrapper.delete_database(db_name)
            print(f"Deleted database {db_name}.")
        print(f"We also created crawler '{crawler_name}'.")
        if Question.ask_question(
```

```
            "Do you want to delete the crawler? (y/n) ", Question.is_yesno
        ):
            wrapper.delete_crawler(crawler_name)
            print(f"Deleted crawler {crawler_name}.")
        print("-" * 88)


def parse_args(args):
    """
    Parse command line arguments.

    :param args: The command line arguments.
    :return: The parsed arguments.
    """
    parser = argparse.ArgumentParser(
        description="Runs the AWS Glue getting started with crawlers and jobs
 scenario. "
        "Before you run this scenario, set up scaffold resources by running "
        "'python scaffold.py deploy'."
    )
    parser.add_argument(
        "role_name",
        help="The name of an IAM role that AWS Glue can assume. This role must
 grant access "
        "to Amazon S3 and to the permissions granted by the AWSGlueServiceRole "
        "managed policy.",
    )
    parser.add_argument(
        "bucket_name",
        help="The name of an S3 bucket that AWS Glue can access to get the job
 script and "
        "put job results.",
    )
    parser.add_argument(
        "--job_script",
        default="flight_etl_job_script.py",
        help="The name of the job script file that is used in the scenario.",
    )
    return parser.parse_args(args)


def main():
    args = parse_args(sys.argv[1:])
    try:
```

```
        print("-" * 88)
        print(
            "Welcome to the AWS Glue getting started with crawlers and jobs
 scenario."
        )
        print("-" * 88)
        scenario = GlueCrawlerJobScenario(
            boto3.client("glue"),
            boto3.resource("iam").Role(args.role_name),
            boto3.resource("s3").Bucket(args.bucket_name),
        )
        scenario.upload_job_script(args.job_script)
        scenario.run(
            "doc-example-crawler",
            "doc-example-database",
            "doc-example-",
            "s3://crawler-public-us-east-1/flight/2016/csv",
            args.job_script,
            "doc-example-job",
        )
        print("-" * 88)
        print(
            "To destroy scaffold resources, including the IAM role and S3 bucket
 "
            "used in this scenario, run 'python scaffold.py destroy'."
        )
        print("\nThanks for watching!")
        print("-" * 88)
    except Exception:
        logging.exception("Something went wrong with the example.")
```

Cree un script ETL que sirva AWS Glue para extraer, transformar y cargar datos durante la ejecución de los trabajos.

```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job
```

```
"""
These custom arguments must be passed as Arguments to the StartJobRun request.
    --input_database    The name of a metadata database that is contained in
 your
                        AWS Glue Data Catalog and that contains tables that
 describe
                        the data to be processed.
    --input_table       The name of a table in the database that describes the
 data to
                        be processed.
    --output_bucket_url An S3 bucket that receives the transformed output data.
"""
args = getResolvedOptions(
    sys.argv, ["JOB_NAME", "input_database", "input_table", "output_bucket_url"]
)
sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args["JOB_NAME"], args)

# Script generated for node S3 Flight Data.
S3FlightData_node1 = glueContext.create_dynamic_frame.from_catalog(
    database=args["input_database"],
    table_name=args["input_table"],
    transformation_ctx="S3FlightData_node1",
)

# This mapping performs two main functions:
# 1. It simplifies the output by removing most of the fields from the data.
# 2. It renames some fields. For example, `fl_date` is renamed to `flight_date`.
ApplyMapping_node2 = ApplyMapping.apply(
    frame=S3FlightData_node1,
    mappings=[
        ("year", "long", "year", "long"),
        ("month", "long", "month", "tinyint"),
        ("day_of_month", "long", "day", "tinyint"),
        ("fl_date", "string", "flight_date", "string"),
        ("carrier", "string", "carrier", "string"),
        ("fl_num", "long", "flight_num", "long"),
        ("origin_city_name", "string", "origin_city_name", "string"),
        ("origin_state_abr", "string", "origin_state_abr", "string"),
        ("dest_city_name", "string", "dest_city_name", "string"),
```

```
        ("dest_state_abr", "string", "dest_state_abr", "string"),
        ("dep_time", "long", "departure_time", "long"),
        ("wheels_off", "long", "wheels_off", "long"),
        ("wheels_on", "long", "wheels_on", "long"),
        ("arr_time", "long", "arrival_time", "long"),
        ("mon", "string", "mon", "string"),
    ],
    transformation_ctx="ApplyMapping_node2",
)

# Script generated for node Revised Flight Data.
RevisedFlightData_node3 = glueContext.write_dynamic_frame.from_options(
    frame=ApplyMapping_node2,
    connection_type="s3",
    format="json",
    connection_options={"path": args["output_bucket_url"], "partitionKeys": []},
    transformation_ctx="RevisedFlightData_node3",
)

job.commit()
```

- Para obtener información sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para Python (Boto3).
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun
  - GetJobRuns
  - GetTables

- ListJobs
- StartCrawler
- StartJobRun

Ruby

SDK para Ruby

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Cree una clase que agrupe AWS Glue las funciones utilizadas en el escenario.

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing
 a simplified interface for common operations.
# It encapsulates the functionality of the AWS SDK for Glue and provides methods
 for interacting with Glue crawlers, databases, tables, jobs, and S3 resources.
# The class initializes with a Glue client and a logger, allowing it to make API
 calls and log any errors or informational messages.
class GlueWrapper
  def initialize(glue_client, logger)
    @glue_client = glue_client
    @logger = logger
  end

  # Retrieves information about a specific crawler.
  #
  # @param name [String] The name of the crawler to retrieve information about.
  # @return [Aws::Glue::Types::Crawler, nil] The crawler object if found, or nil
 if not found.
  def get_crawler(name)
    @glue_client.get_crawler(name: name)
  rescue Aws::Glue::Errors::EntityNotFoundException
    @logger.info("Crawler #{name} doesn't exist.")
    false
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get crawler #{name}: \n#{e.message}")
```

```
    raise
  end

  # Creates a new crawler with the specified configuration.
  #
  # @param name [String] The name of the crawler.
  # @param role_arn [String] The ARN of the IAM role to be used by the crawler.
  # @param db_name [String] The name of the database where the crawler stores its
metadata.
  # @param db_prefix [String] The prefix to be added to the names of tables that
the crawler creates.
  # @param s3_target [String] The S3 path that the crawler will crawl.
  # @return [void]
  def create_crawler(name, role_arn, db_name, db_prefix, s3_target)
    @glue_client.create_crawler(
      name: name,
      role: role_arn,
      database_name: db_name,
      targets: {
        s3_targets: [
          {
            path: s3_target
          }
        ]
      }
    )
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not create crawler: \n#{e.message}")
    raise
  end

  # Starts a crawler with the specified name.
  #
  # @param name [String] The name of the crawler to start.
  # @return [void]
  def start_crawler(name)
    @glue_client.start_crawler(name: name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not start crawler #{name}: \n#{e.message}")
    raise
  end

  # Deletes a crawler with the specified name.
  #
```

```
  # @param name [String] The name of the crawler to delete.
  # @return [void]
  def delete_crawler(name)
    @glue_client.delete_crawler(name: name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete crawler #{name}: \n#{e.message}")
    raise
  end

  # Retrieves information about a specific database.
  #
  # @param name [String] The name of the database to retrieve information about.
  # @return [Aws::Glue::Types::Database, nil] The database object if found, or
 nil if not found.
  def get_database(name)
    response = @glue_client.get_database(name: name)
    response.database
rescue Aws::Glue::Errors::GlueException => e
  @logger.error("Glue could not get database #{name}: \n#{e.message}")
  raise
  end

  # Retrieves a list of tables in the specified database.
  #
  # @param db_name [String] The name of the database to retrieve tables from.
  # @return [Array<Aws::Glue::Types::Table>]
  def get_tables(db_name)
    response = @glue_client.get_tables(database_name: db_name)
    response.table_list
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get tables #{db_name}: \n#{e.message}")
    raise
  end

  # Creates a new job with the specified configuration.
  #
  # @param name [String] The name of the job.
  # @param description [String] The description of the job.
  # @param role_arn [String] The ARN of the IAM role to be used by the job.
  # @param script_location [String] The location of the ETL script for the job.
  # @return [void]
  def create_job(name, description, role_arn, script_location)
    @glue_client.create_job(
      name: name,
```

```
      description: description,
      role: role_arn,
      command: {
        name: "glueetl",
        script_location: script_location,
        python_version: "3"
      },
      glue_version: "3.0"
    )
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not create job #{name}: \n#{e.message}")
    raise
  end

  # Starts a job run for the specified job.
  #
  # @param name [String] The name of the job to start the run for.
  # @param input_database [String] The name of the input database for the job.
  # @param input_table [String] The name of the input table for the job.
  # @param output_bucket_name [String] The name of the output S3 bucket for the
job.
  # @return [String] The ID of the started job run.
  def start_job_run(name, input_database, input_table, output_bucket_name)
    response = @glue_client.start_job_run(
      job_name: name,
      arguments: {
        '--input_database': input_database,
        '--input_table': input_table,
        '--output_bucket_url': "s3://#{output_bucket_name}/"
      }
    )
    response.job_run_id
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not start job run #{name}: \n#{e.message}")
    raise
  end

  # Retrieves a list of jobs in AWS Glue.
  #
  # @return [Aws::Glue::Types::ListJobsResponse]
  def list_jobs
    @glue_client.list_jobs
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not list jobs: \n#{e.message}")
```

```
    raise
  end

  # Retrieves a list of job runs for the specified job.
  #
  # @param job_name [String] The name of the job to retrieve job runs for.
  # @return [Array<Aws::Glue::Types::JobRun>]
  def get_job_runs(job_name)
    response = @glue_client.get_job_runs(job_name: job_name)
    response.job_runs
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get job runs: \n#{e.message}")
  end

  # Retrieves data for a specific job run.
  #
  # @param job_name [String] The name of the job run to retrieve data for.
  # @return [Glue::Types::GetJobRunResponse]
  def get_job_run(job_name, run_id)
    @glue_client.get_job_run(job_name: job_name, run_id: run_id)
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not get job runs: \n#{e.message}")
  end

  # Deletes a job with the specified name.
  #
  # @param job_name [String] The name of the job to delete.
  # @return [void]
  def delete_job(job_name)
    @glue_client.delete_job(job_name: job_name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete job: \n#{e.message}")
  end

  # Deletes a table with the specified name.
  #
  # @param database_name [String] The name of the catalog database in which the
table resides.
  # @param table_name [String] The name of the table to be deleted.
  # @return [void]
  def delete_table(database_name, table_name)
    @glue_client.delete_table(database_name: database_name, name: table_name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete job: \n#{e.message}")
```

```
  end

  # Removes a specified database from a Data Catalog.
  #
  # @param database_name [String] The name of the database to delete.
  # @return [void]
  def delete_database(database_name)
    @glue_client.delete_database(name: database_name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete database: \n#{e.message}")
  end

  # Uploads a job script file to an S3 bucket.
  #
  # @param file_path [String] The local path of the job script file.
  # @param bucket_resource [Aws::S3::Bucket] The S3 bucket resource to upload the
 file to.
  # @return [void]
  def upload_job_script(file_path, bucket_resource)
    File.open(file_path) do |file|
      bucket_resource.client.put_object({
       body: file,
       bucket: bucket_resource.name,
       key: file_path
     })
    end
  rescue Aws::S3::Errors::S3UploadFailedError => e
    @logger.error("S3 could not upload job script: \n#{e.message}")
    raise
  end

end
```

Crear una clase que ejecute el escenario.

```
class GlueCrawlerJobScenario
  def initialize(glue_client, glue_service_role, glue_bucket, logger)
    @glue_client = glue_client
    @glue_service_role = glue_service_role
    @glue_bucket = glue_bucket
    @logger = logger
  end
```

```
 def run(crawler_name, db_name, db_prefix, data_source, job_script, job_name)
   wrapper = GlueWrapper.new(@glue_client, @logger)

   new_step(1, "Create a crawler")
   puts "Checking for crawler #{crawler_name}."
   crawler = wrapper.get_crawler(crawler_name)
   if crawler == false
     puts "Creating crawler #{crawler_name}."
     wrapper.create_crawler(crawler_name, @glue_service_role.arn, db_name,
db_prefix, data_source)
     puts "Successfully created #{crawler_name}:"
     crawler = wrapper.get_crawler(crawler_name)
     puts JSON.pretty_generate(crawler).yellow
   end
   print "\nDone!\n".green

   new_step(2, "Run a crawler to output a database.")
   puts "Location of input data analyzed by crawler: #{data_source}"
   puts "Outputs: a Data Catalog database in CSV format containing metadata on
input."
   wrapper.start_crawler(crawler_name)
   puts "Starting crawler... (this typically takes a few minutes)"
   crawler_state = nil
   while crawler_state != "READY"
     custom_wait(15)
     crawler = wrapper.get_crawler(crawler_name)
     crawler_state = crawler[0]["state"]
     print "Status check: #{crawler_state}.".yellow
   end
   print "\nDone!\n".green

   new_step(3, "Query the database.")
   database = wrapper.get_database(db_name)
   puts "The crawler created database #{db_name}:"
   print "#{database}".yellow
   puts "\nThe database contains these tables:"
   tables = wrapper.get_tables(db_name)
   tables.each_with_index do |table, index|
     print "\t#{index + 1}. #{table['name']}".yellow
   end
   print "\nDone!\n".green

   new_step(4, "Create a job definition that runs an ETL script.")
```

```
    puts "Uploading Python ETL script to S3..."
    wrapper.upload_job_script(job_script, @glue_bucket)
    puts "Creating job definition #{job_name}:\n"
    response = wrapper.create_job(job_name, "Getting started example job.",
@glue_service_role.arn, "s3://#{@glue_bucket.name}/#{job_script}")
    puts JSON.pretty_generate(response).yellow
    print "\nDone!\n".green

    new_step(5, "Start a new job")
    job_run_status = nil
    job_run_id = wrapper.start_job_run(
      job_name,
      db_name,
      tables[0]["name"],
      @glue_bucket.name
    )
    puts "Job #{job_name} started. Let's wait for it to run."
    until ["SUCCEEDED", "STOPPED", "FAILED", "TIMEOUT"].include?(job_run_status)
      custom_wait(10)
      job_run = wrapper.get_job_runs(job_name)
      job_run_status = job_run[0]["job_run_state"]
      print "Status check: #{job_name}/#{job_run_id} - #{job_run_status}.".yellow
    end
    print "\nDone!\n".green

    new_step(6, "View results from a successful job run.")
    if job_run_status == "SUCCEEDED"
      puts "Data from your job run is stored in your S3 bucket
'#{@glue_bucket.name}'. Files include:"
      begin

        # Print the key name of each object in the bucket.
        @glue_bucket.objects.each do |object_summary|
          if object_summary.key.include?("run-")
            print "#{object_summary.key}".yellow
          end
        end

        # Print the first 256 bytes of a run file
        desired_sample_objects = 1
        @glue_bucket.objects.each do |object_summary|
          if object_summary.key.include?("run-")
            if desired_sample_objects > 0
              sample_object = @glue_bucket.object(object_summary.key)
```

```
              sample = sample_object.get(range: "bytes=0-255").body.read
              puts "\nSample run file contents:"
              print "#{sample}".yellow
              desired_sample_objects -= 1
            end
          end
        end
      rescue Aws::S3::Errors::ServiceError => e
        logger.error(
          "Couldn't get job run data. Here's why: %s: %s",
          e.response.error.code, e.response.error.message
        )
        raise
      end
    end
    print "\nDone!\n".green

    new_step(7, "Delete job definition and crawler.")
    wrapper.delete_job(job_name)
    puts "Job deleted: #{job_name}."
    wrapper.delete_crawler(crawler_name)
    puts "Crawler deleted: #{crawler_name}."
    wrapper.delete_table(db_name, tables[0]["name"])
    puts "Table deleted: #{tables[0]["name"]} in #{db_name}."
    wrapper.delete_database(db_name)
    puts "Database deleted: #{db_name}."
    print "\nDone!\n".green
  end
end

def main

  banner("../../helpers/banner.txt")
  puts
 "#################################################################################################
  puts "#
                            #".yellow
  puts "#                                          EXAMPLE CODE DEMO:
                            #".yellow
  puts "#                                              AWS Glue
                            #".yellow
  puts "#
                            #".yellow
```

```
 puts
"#########################################################################################
 puts ""
 puts "You have launched a demo of AWS Glue using the AWS for Ruby v3 SDK. Over
the next 60 seconds, it will"
 puts "do the following:"
 puts "    1. Create a crawler."
 puts "    2. Run a crawler to output a database."
 puts "    3. Query the database."
 puts "    4. Create a job definition that runs an ETL script."
 puts "    5. Start a new job."
 puts "    6. View results from a successful job run."
 puts "    7. Delete job definition and crawler."
 puts ""

 confirm_begin
 billing
 security
 puts "\e[H\e[2J"

 # Set input file names
 job_script_filepath = "job_script.py"
 resource_names = YAML.load_file("resource_names.yaml")

 # Instantiate existing IAM role.
 iam = Aws::IAM::Resource.new(region: "us-east-1")
 iam_role_name = resource_names["glue_service_role"]
 iam_role = iam.role(iam_role_name)

 # Instantiate existing S3 bucket.
 s3 = Aws::S3::Resource.new(region: "us-east-1")
 s3_bucket_name = resource_names["glue_bucket"]
 s3_bucket = s3.bucket(s3_bucket_name)

 scenario = GlueCrawlerJobScenario.new(
   Aws::Glue::Client.new(region: "us-east-1"),
   iam_role,
   s3_bucket,
   @logger
 )

 random_int = rand(10 ** 4)
 scenario.run(
   "doc-example-crawler-#{random_int}",
```

```
    "doc-example-database-#{random_int}",
    "doc-example-#{random_int}-",
    "s3://crawler-public-us-east-1/flight/2016/csv",
    job_script_filepath,
    "doc-example-job-#{random_int}"
  )

  puts "-" * 88
  puts "You have reached the end of this tour of AWS Glue."
  puts "To destroy CDK-created resources, run:\n       cdk destroy"
  puts "-" * 88

end
```

Cree un script ETL que sirva AWS Glue para extraer, transformar y cargar datos durante la ejecución de los trabajos.

```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job

"""
These custom arguments must be passed as Arguments to the StartJobRun request.
    --input_database    The name of a metadata database that is contained in
 your
                        AWS Glue Data Catalog and that contains tables that
 describe
                        the data to be processed.
    --input_table       The name of a table in the database that describes the
 data to
                        be processed.
    --output_bucket_url An S3 bucket that receives the transformed output data.
"""
args = getResolvedOptions(
    sys.argv, ["JOB_NAME", "input_database", "input_table", "output_bucket_url"]
)
sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
```

```
job = Job(glueContext)
job.init(args["JOB_NAME"], args)

# Script generated for node S3 Flight Data.
S3FlightData_node1 = glueContext.create_dynamic_frame.from_catalog(
    database=args["input_database"],
    table_name=args["input_table"],
    transformation_ctx="S3FlightData_node1",
)

# This mapping performs two main functions:
# 1. It simplifies the output by removing most of the fields from the data.
# 2. It renames some fields. For example, `fl_date` is renamed to `flight_date`.
ApplyMapping_node2 = ApplyMapping.apply(
    frame=S3FlightData_node1,
    mappings=[
        ("year", "long", "year", "long"),
        ("month", "long", "month", "tinyint"),
        ("day_of_month", "long", "day", "tinyint"),
        ("fl_date", "string", "flight_date", "string"),
        ("carrier", "string", "carrier", "string"),
        ("fl_num", "long", "flight_num", "long"),
        ("origin_city_name", "string", "origin_city_name", "string"),
        ("origin_state_abr", "string", "origin_state_abr", "string"),
        ("dest_city_name", "string", "dest_city_name", "string"),
        ("dest_state_abr", "string", "dest_state_abr", "string"),
        ("dep_time", "long", "departure_time", "long"),
        ("wheels_off", "long", "wheels_off", "long"),
        ("wheels_on", "long", "wheels_on", "long"),
        ("arr_time", "long", "arrival_time", "long"),
        ("mon", "string", "mon", "string"),
    ],
    transformation_ctx="ApplyMapping_node2",
)

# Script generated for node Revised Flight Data.
RevisedFlightData_node3 = glueContext.write_dynamic_frame.from_options(
    frame=ApplyMapping_node2,
    connection_type="s3",
    format="json",
    connection_options={"path": args["output_bucket_url"], "partitionKeys": []},
    transformation_ctx="RevisedFlightData_node3",
)
```

```
job.commit()
```

- Para obtener información sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK for Ruby .
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun
  - GetJobRuns
  - GetTables
  - ListJobs
  - StartCrawler
  - StartJobRun

Rust

SDK para Rust

> **Note**
>
> Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el Repositorio de ejemplos de código de AWS.

Cree y ejecute un rastreador que rastree un bucket público de Amazon Simple Storage Service (Amazon S3) y genere una base de metadatos que describa los datos con formato CSV que encuentra.

```
        let create_crawler = glue
            .create_crawler()
            .name(self.crawler())
            .database_name(self.database())
            .role(self.iam_role.expose_secret())
            .targets(
                CrawlerTargets::builder()
                    .s3_targets(S3Target::builder().path(CRAWLER_TARGET).build())
                    .build(),
            )
            .send()
            .await;

        match create_crawler {
            Err(err) => {
                let glue_err: aws_sdk_glue::Error = err.into();
                match glue_err {
                    aws_sdk_glue::Error::AlreadyExistsException(_) => {
                        info!("Using existing crawler");
                        Ok(())
                    }
                    _ => Err(GlueMvpError::GlueSdk(glue_err)),
                }
            }
            Ok(_) => Ok(()),
        }?;

        let start_crawler =
glue.start_crawler().name(self.crawler()).send().await;

        match start_crawler {
            Ok(_) => Ok(()),
            Err(err) => {
                let glue_err: aws_sdk_glue::Error = err.into();
                match glue_err {
                    aws_sdk_glue::Error::CrawlerRunningException(_) => Ok(()),
                    _ => Err(GlueMvpError::GlueSdk(glue_err)),
                }
            }
```

```
        }?;
```

Enumere información sobre bases de datos y tablas en su AWS Glue Data Catalog.

```
        let database = glue
            .get_database()
            .name(self.database())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?
            .to_owned();
        let database = database
            .database()
            .ok_or_else(|| GlueMvpError::Unknown("Could not find
 database".into()))?;

        let tables = glue
            .get_tables()
            .database_name(self.database())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;

        let tables = tables.table_list();
```

Cree y ejecute un trabajo que extraiga datos CSV del bucket de Amazon S3 de origen, los transforme quitando campos y cambiándoles el nombre, y cargue el resultado con formato JSON en otro bucket de Amazon S3.

```
        let create_job = glue
            .create_job()
            .name(self.job())
            .role(self.iam_role.expose_secret())
            .command(
                JobCommand::builder()
                    .name("glueetl")
                    .python_version("3")
                    .script_location(format!("s3://{}/job.py", self.bucket()))
                    .build(),
            )
```

```
            .glue_version("3.0")
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;

        let job_name = create_job.name().ok_or_else(|| {
            GlueMvpError::Unknown("Did not get job name after creating
job".into())
        })?;

        let job_run_output = glue
            .start_job_run()
            .job_name(self.job())
            .arguments("--input_database", self.database())
            .arguments(
                "--input_table",
                self.tables
                    .first()
                    .ok_or_else(|| GlueMvpError::Unknown("Missing crawler
table".into()))?
                    .name(),
            )
            .arguments("--output_bucket_url", self.bucket())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;

        let job = job_run_output
            .job_run_id()
            .ok_or_else(|| GlueMvpError::Unknown("Missing run id from just
started job".into()))?
            .to_string();
```

Elimine todos los recursos creados en la demostración.

```
        glue.delete_job()
            .job_name(self.job())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;

        for t in &self.tables {
```

```
            glue.delete_table()
                .name(t.name())
                .database_name(self.database())
                .send()
                .await
                .map_err(GlueMvpError::from_glue_sdk)?;
        }

        glue.delete_database()
            .name(self.database())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;

        glue.delete_crawler()
            .name(self.crawler())
            .send()
            .await
            .map_err(GlueMvpError::from_glue_sdk)?;
```

- Para obtener información sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para Rust.
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun
  - GetJobRuns
  - GetTables
  - ListJobs

- StartCrawler
- StartJobRun

Para obtener una lista completa de las guías para desarrolladores del AWS SDK y ejemplos de código, consulteUso de este servicio con un AWS SDK. En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Seguridad en AWS Glue

La seguridad en la nube de AWS es la máxima prioridad. Como cliente de AWS, se beneficia de una arquitectura de red y un centro de datos que se han diseñado para satisfacer los requisitos de seguridad de las organizaciones más exigentes.

La seguridad es una responsabilidad compartida entre AWS y el usuario. El modelo de responsabilidad compartida la describe como seguridad de la nube y seguridad en la nube:

- Seguridad de la nube: AWS es responsable de proteger la infraestructura que ejecuta los servicios de AWS en la nube de AWS. AWS también proporciona servicios que puede utilizar de forma segura. Auditores externos prueban y verifican periódicamente la eficacia de nuestra seguridad en el marco de los programas de conformidad de AWS. Para obtener más información sobre los programas de conformidad que se aplican a AWS Glue, consulte Servicios de AWS en el ámbito del programa de conformidad.
- Seguridad en la nube: su responsabilidad se determina según el servicio de AWS que utilice. También es responsable de otros factores, incluida la confidencialidad de los datos, los requisitos de la empresa y la legislación y la normativa aplicables.

Esta documentación le ayuda a comprender cómo aplicar el modelo de responsabilidad compartida a la hora de utilizar AWS Glue. Los siguientes temas le mostrarán cómo configurar AWS Glue para satisfacer sus objetivos de seguridad y de conformidad. También puede aprender a utilizar otros servicios de AWS que lo ayuden a monitorizar y proteger los recursos de AWS Glue.

Temas

- Protección de los datos en AWS Glue
- Gestión de identidad y acceso para AWS Glue
- Registro y monitorización en AWS Glue
- Validación de conformidad para AWS Glue
- Resiliencia en AWS Glue
- Seguridad de la infraestructura en AWS Glue

Protección de los datos en AWS Glue

AWS Glue ofrece varias características que están diseñadas para ayudar a proteger sus datos.

Temas

- Cifrado en reposo
- Cifrado en tránsito
- Conformidad con FIPS
- Administración de claves
- Dependencia de AWS Glue en otros servicios de AWS
- Puntos de conexión de desarrollo

Cifrado en reposo

AWS Glue admite el cifrado de datos en reposo para Creación de trabajos de ETL visuales con AWS Glue Studio y Desarrollo de scripts usando puntos de conexión de desarrollo. Puede configurar trabajos de extracción, transformación y carga (ETL) y puntos de enlace de desarrollo para utilizar claves de AWS Key Management Service (AWS KMS) para escribir datos cifrados en reposo. También puede cifrar los metadatos almacenados en AWS Glue Data Catalog con claves que administre con AWS KMS. Además, puede usar las claves de AWS KMS para cifrar marcadores de trabajo y los registros que generan los rastreadores y los trabajos de ETL.

Puede cifrar objetos de metadatos en su AWS Glue Data Catalog además de los datos escritos en Amazon Simple Storage Service (Amazon S3) y Amazon CloudWatch Logs por trabajos, rastreadores y puntos de enlace de desarrollo. Cuando cree trabajos, rastreadores y puntos de enlace de desarrollo en AWS Glue, puede proporcionar valores de cifrado al asociar una configuración de seguridad. Las configuraciones de seguridad contienen claves de cifrado del servidor administradas por Amazon S3 (SSE-S3) o claves maestras de cliente (CMK) almacenadas en AWS KMS (SSE-KMS). Puede crear configuraciones de seguridad a través de la consola de AWS Glue.

También puede habilitar el cifrado de todo el Catálogo de datos en su cuenta. Para ello, especifique las CMK almacenadas en AWS KMS.

> **Important**
>
> AWS Glue solo es compatible con las claves simétricas administradas por el cliente. Para obtener más información, consulte Claves administradas por el cliente (CMK) en la Guía para desarrolladores de AWS Key Management Service.

Si el cifrado está activado, al agregar objetos del Catálogo de datos, ejecutar rastreadores, ejecutar trabajos o iniciar puntos de enlace de desarrollo, se utilizan claves SSE-S3 o SSE-KMS para escribir datos en reposo. Además, puede configurar AWS Glue para que solo acceda a almacenamiento de datos de Java Database Connectivity (JDBC) a través de un protocolo de seguridad de la capa de transporte (TLS) de confianza.

En AWS Glue, puede controlar la configuración de cifrado en los siguientes lugares:

- La configuración del Catálogo de datos.
- Las configuraciones de seguridad que cree.
- Configuración del cifrado del lado del servidor (SSE-S3 o SSE-KMS) que se transfiere como parámetro a su trabajo de ETL (extracción, transformación y carga) de AWS Glue.

Para obtener más información acerca de cómo configurar el cifrado, consulte Configuración del cifrado en AWS Glue.

Temas

- Cifrado del Catálogo de datos
- Cifrado de las contraseñas de conexión
- Cifrado de datos escritos por AWS Glue

Cifrado del Catálogo de datos

El cifrado AWS Glue Data Catalog permite una seguridad mejorada para los datos confidenciales. AWS Glue se integra con AWS Key Management Service (AWS KMS) para cifrar los metadatos almacenados en el Catálogo de datos. Las configuraciones de cifrado para los recursos en el Catálogo de datos se pueden activar o desactivar a través de la consola de AWS Glue o el AWS CLI.

Si el cifrado está activado en el Catálogo de datos, se cifrarán todos los objetos nuevos que se creen. Si el cifrado está desactivado, no se cifrarán los objetos nuevos que se creen, pero los objetos que ya estén cifrados permanecerán cifrados.

Se puede cifrar el Catálogo de datos en su totalidad con las claves de cifrado administradas por AWS o con las claves de cifrado administradas por el cliente. Para obtener más información sobre los tipos y los estados de las claves, consulte Conceptos de AWS Key Management Service en la Guía para desarrolladores de AWS Key Management Service.

Claves administradas por AWS

Las claves administradas por AWS son claves KMS que aparecen en su cuenta y que un servicio de AWS integrado con AWS KMS crea, administra y utiliza en su nombre. Las claves administradas por AWS en su cuenta se pueden visualizar, junto con sus políticas de claves, y en los registros de AWS CloudTrail se puede auditar su utilización. Sin embargo, estas claves no se pueden administrar ni cambiar sus permisos.

El cifrado en reposo se integra de manera automática con AWS KMS para administrar las claves administradas por AWS para AWS Glue y que se utilizan para cifrar los metadatos. Si no existe una clave administrada por AWS cuando se active el cifrado de los metadatos, AWS KMS creará una nueva de manera automática.

Para obtener más información, consulte Claves administradas de AWS.

Claves administradas por el cliente

Las claves administradas por el cliente son claves KMS de su Cuenta de AWS, que usted ha creado, posee y administra. Usted tiene el control total sobre estas claves de KMS. Puede hacer lo siguiente:

- Establecer y mantener las políticas de claves, las políticas de IAM y las subvenciones
- Habilitar y deshabilitar las claves
- Rotar su material de cifrado
- Agregue etiquetas
- Crear alias que hagan referencia a las claves
- Programar la eliminación de las claves

Para obtener más información sobre la administración de los permisos de una clave administrada por el cliente, consulte Claves administradas por el cliente.

> **Important**
>
> AWS Glue solo es compatible con las claves simétricas administradas por el cliente. La lista de claves de KMS solo muestra las claves simétricas. Sin embargo, si selecciona Seleccionar un ARN de clave de KMS, la consola le deja escribir un ARN para cualquier tipo de clave. Asegúrese de introducir sólo ARN para claves simétricas.

Para crear una clave simétrica administrada por el cliente, siga las instrucciones para la creación de claves simétricas administradas por el cliente en la Guía para desarrolladores de AWS Key Management Service.

Al activar el cifrado del Catálogo de datos en reposo, se cifran con claves MKS los tipos de recursos que se mencionan a continuación:

- Bases de datos
- Tablas
- Particiones
- Versiones de tablas
- Estadísticas de las columnas
- Funciones definidas por el usuario
- Vistas del Catálogo de datos

Contexto de cifrado de AWS Glue

Un contexto de cifrado es un conjunto de pares de valor de clave opcional que pueden contener información contextual adicional sobre los datos. AWS KMS utiliza el contexto de cifrado como información autenticada adicional para permitir el cifrado autenticado. Cuando se incluye un contexto de cifrado en una solicitud para cifrar datos, AWS KMS vincula el contexto de cifrado a los datos cifrados. Para cifrar los datos, se incluye el mismo contexto de cifrado en la solicitud. AWS Glue utiliza el mismo contexto de cifrado en todas las operaciones de cifrado de AWS KMS, en las cuales `glue_catalog_id` es la clave y el valor es `catalogId`.

```
"encryptionContext": {
    "glue_catalog_id": "111122223333"
}
```

Al utilizar una clave administrada por AWS o una clave simétrica administrada por el cliente para cifrar el Catálogo de datos, también se puede utilizar un contexto de cifrado en los documentos de la auditoría y en los registros para identificar la clave que se utiliza. El contexto de cifrado también aparece en los registros que generan los registros de AWS CloudTrail o de Amazon CloudWatch.

Habilitación del cifrado

El cifrado de los objetos de AWS Glue Data Catalog se puede activar en Configuraciones del Catálogo de datos en la consola de AWS Glue o con la AWS CLI.

Console

Para habilitar el cifrado mediante la consola

1. Inicie sesión en la AWS Management Console y abra la consola de AWS Glue en https://console.aws.amazon.com/glue/.
2. Seleccione Catálogo de datos en el panel de navegación.
3. En la página Configuraciones del Catálogo de datos, seleccione la casilla Cifrado de metadatos y luego seleccione una clave de AWS KMS.

   Al activar el cifrado, y si no se especificó una clave administrada por el cliente, las configuraciones de cifrado utilizan una clave de KMS administrada por AWS.
4. (Opcional) Al utilizar una clave administrada por el cliente para cifrar el Catálogo de datos, el Catálogo ofrece una opción para registrar un rol de IAM para cifrar y descifrar los recursos. Debe conceder permisos de rol de IAM para que los asuma AWS Glue en su nombre. Esto incluye los permisos de AWS KMS para cifrar y descifrar datos.

   Al crear un recurso nuevo en el Catálogo de datos, AWS Glue asume el rol de IAM que se brindó para cifrar los datos. De forma similar, cuando un cliente accede al recurso, AWS Glue asume el rol de IAM para cifrar los datos. Si se registra un rol de IAM con los permisos necesarios, la entidad principal que realiza la llamada ya no necesita permisos para acceder a la clave y descifrar los datos.

   > **Important**
   >
   > Las operaciones de KMS se pueden delegar a un rol de IAM solo cuando se utiliza una clave administrada por el cliente para cifrar los recursos del Catálogo de datos. La característica de delegación de roles de KMS no es compatible con la utilización de claves administradas por AWS para el cifrado de los recursos del Catálogo de datos en este momento.

> **Warning**
>
> Al activar un rol de IAM para delegar las operaciones de KMS, ya no podrá acceder a los recursos del Catálogo de datos cifrados anteriormente con una clave administrada por AWS.

a. Para habilitar un rol de IAM que AWS Glue pueda asumir para cifrar y descifrar los datos en su nombre, seleccione la opción Delegar operaciones de KMS a un rol de IAM.

b. Luego, seleccione un rol de IAM.

   Para crear un rol de IAM;, consulte Crear un rol de IAM para AWS Glue.

   El rol de IAM asumido por AWS Glue para acceder al Catálogo de datos debe contar con los permisos para cifrar y descifrar los metadatos en el Catálogo de datos. Puede crear un rol de IAM y adjuntar las políticas integradas que me enumeran a continuación:

   - La política que figura a continuación se puede agregar para incluir los permisos de AWS KMS para cifrar y descifrar el Catálogo de datos.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:Encrypt",
                "kms:GenerateDataKey"
            ],
            "Resource": "arn:aws:kms:<region>:<account-id>:key/<key-id>"
        }
    ]
}
```

   - Luego, agregue al rol la política a continuación para que el servicio de AWS Glue asuma el rol de IAM.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "Service": "glue.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
 }
```

- Luego, agregue el permiso de `iam:PassRole` al rol de IAM.

```
                    {
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::<account-id>:role/<encryption-role-name>"
            ]
        }
    ]
}
```

Al activar el cifrado, y si no se especificó un rol de IAM para que AWS Glue asuma, la entidad principal que acceda al Catálogo de datos debe contar con los permisos para realizar las operaciones de API que se enumeran a continuación:

- `kms:Decrypt`
- `kms:Encrypt`
- `kms:GenerateDataKey`

AWS CLI

Para habilitar el cifrado mediante el SDK o AWS CLI

- Use la operación `PutDataCatalogEncryptionSettings` de la API. Si no se especifica ninguna clave, AWS Glue utiliza una clave de cifrado administrada por AWS para la cuenta del cliente para cifrar el Catálogo de datos.

```
aws glue put-data-catalog-encryption-settings \
  --data-catalog-encryption-settings '{
      "EncryptionAtRest": {
      "CatalogEncryptionMode": "SSE-KMS-WITH-SERVICE-ROLE",
      "SseAwsKmsKeyId": "arn:aws:kms:<region>:<account-id>:key/<key-id>",
      "CatalogEncryptionServiceRole":"arn:aws:iam::<account-
id>:role/<encryption-role-name>"
    }

  }'
```

Al activar el cifrado, se cifran todos los objetos que se creen en el Catálogo de datos. Si elimina la configuración, ya no se cifrarán los objetos que se creen en el Catálogo. Puede seguir accediendo a los objetos que ya están cifrados en el Catálogo de datos con los permisos de KMS requeridos.

> **Important**
>
> La clave AWS KMS debe permanecer disponible en el almacén de claves AWS KMS para todos los objetos que se cifran con ella en el Catálogo de datos. Si elimina la clave, los objetos ya no se podrán descifrar. Es posible que en determinados casos le interese esta opción para evitar el acceso a los metadatos del Catálogo de datos.

Monitoreo de las claves de KMS para AWS Glue

Al utilizar las claves de KMS para los recursos del Catálogo de datos, se pueden utilizar los Registros de AWS CloudTrail o de Amazon CloudWatch para rastrear las solicitudes que envía AWS Glue a AWS KMS. AWS CloudTrail monitorea y registra las operaciones de KMS a las cuales AWS Glue llama para acceder a los datos cifrados con las claves de KMS.

Los ejemplos que se muestran a continuación son eventos de AWS CloudTrail para las operaciones de `Decrypt` y `GenerateDataKey`.

Decrypt


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAXPHTESTANDEXAMPLE:Sampleuser01",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/Sampleuser01",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAXPHTESTANDEXAMPLE",
                "arn": "arn:aws:iam::111122223333:role/Admin",
                "accountId": "111122223333",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2024-01-10T14:33:56Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "glue.amazonaws.com"
    },
    "eventTime": "2024-01-10T15:18:11Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "eu-west-2",
    "sourceIPAddress": "glue.amazonaws.com",
    "userAgent": "glue.amazonaws.com",
    "requestParameters": {
        "encryptionContext": {
            "glue_catalog_id": "111122223333"
        },
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT"
    },
    "responseElements": null,
    "requestID": "43b019aa-34b8-4798-9b98-ee968b2d63df",
```

```
    "eventID": "d7614763-d3fe-4f84-a1e1-3ca4d2a5bbd5",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:<region>:111122223333:key/<key-id>"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management",
    "sessionCredentialFromConsole": "true"
}
```

GenerateDataKey

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId":
 "AROAXPHTESTANDEXAMPLE:V_00_GLUE_KMS_GENERATE_DATA_KEY_111122223333",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/
V_00_GLUE_KMS_GENERATE_DATA_KEY_111122223333",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAXPHTESTANDEXAMPLE",
                "arn": "arn:aws:iam::111122223333:role/Admin",
                "accountId": "AKIAIOSFODNN7EXAMPLE",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2024-01-05T21:15:47Z",
                "mfaAuthenticated": "false"
            }
        },
```

```
        "invokedBy": "glue.amazonaws.com"
    },
    "eventTime": "2024-01-05T21:15:47Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKey",
    "awsRegion": "eu-west-2",
    "sourceIPAddress": "glue.amazonaws.com",
    "userAgent": "glue.amazonaws.com",
    "requestParameters": {
        "keyId": "arn:aws:kms:eu-west-2:AKIAIOSFODNN7EXAMPLE:key/
AKIAIOSFODNN7EXAMPLE",
        "encryptionContext": {
            "glue_catalog_id": "111122223333"
        },
        "keySpec": "AES_256"
    },
    "responseElements": null,
    "requestID": "64d1783a-4b62-44ba-b0ab-388b50188070",
    "eventID": "1c73689b-2ef2-443b-aed7-8c126585ca5e",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:eu-west-2:111122223333:key/AKIAIOSFODNN7EXAMPLE"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

Cifrado de las contraseñas de conexión

Puede recuperar las contraseñas de conexión en AWS Glue Data Catalog mediante las operaciones de la API `GetConnection` y `GetConnections`. Estas contraseñas se almacenan en la conexión del Catálogo de datos y se utilizan cuando AWS Glue se conecta a un almacén de datos de Java Database Connectivity (JDBC). Cuando se creó o actualizó la conexión, una opción en la configuración del Catálogo de datos determinó si la contraseña estaba cifrada, y en caso afirmativo, qué clave AWS Key Management Service (AWS KMS) se especificó.

En la consola de AWS Glue, puede habilitar esta opción en la página Data catalog settings (Configuración del Catálogo de datos).

Para cifrar las contraseñas de conexión

1. Inicie sesión en AWS Management Console y abra la consola de AWS Glue en https://console.aws.amazon.com/glue/.
2. Seleccione Settings (Configuración) en el panel de navegación.
3. En la página Data catalog settings (Configuración del Catálogo de datos), seleccione Encrypt connection passwords (Cifrar contraseñas de conexión) y elija una clave AWS KMS.

   > **Important**
   >
   > AWS Glue solo soporta claves maestras de cliente (CMK) simétricas. La lista de AWS KMS key (Clave KMS) muestra únicamente claves simétricas. Sin embargo, si selecciona Choose a AWS KMS key ARN (Elegir un ARN de clave KMS), la consola le permite introducir un ARN para cualquier tipo de clave. Asegúrese de introducir sólo ARN para claves simétricas.

   Para obtener más información, consulte Configuración del Catálogo de datos .

Cifrado de datos escritos por AWS Glue

Una configuración de seguridad es un conjunto de propiedades de seguridad que AWS Glue puede usar. Puede utilizar una configuración de seguridad para cifrar los datos en reposo. En las siguientes situaciones se muestran algunas de las maneras en las que puede utilizar una configuración de seguridad.

- Asocie una configuración de seguridad a un rastreador de AWS Glue para escribir registros cifrados de Amazon CloudWatch Logs. Para obtener más información sobre cómo asociar las configuraciones de seguridad a los rastreadores, consulte the section called “Paso 3: Establecer configuración de seguridad”.
- Asocie una configuración de seguridad a un trabajo de extracción, transformación y carga (ETL) para escribir destinos de Amazon Simple Storage Service (Amazon S3) cifrados y registros de CloudWatch Logs cifrados.
- Asocie una configuración de seguridad a un trabajo de ETL para escribir sus marcadores de trabajo como datos de Amazon S3 cifrados.
- Asocie una configuración de seguridad a un punto de enlace de desarrollo para escribir destinos de Amazon S3 cifrados.

> **Important**
>
> En la actualidad, una configuración de seguridad reemplaza cualquier configuración de cifrado del lado del servidor (SSE-S3) que se haya pasado como un parámetro de flujo de trabajo de ETL. Por lo tanto, si un flujo de trabajo tiene asociados una configuración de seguridad y un parámetro de SSE-S3, este último se ignorará.

Para obtener más información acerca de las configuraciones de seguridad, consulte Trabajar con configuraciones de seguridad en la consola de AWS Glue.

Temas

- Configuración de AWS Glue para utilizar configuraciones de seguridad
- Creación de una ruta a AWS KMS para rastreadores y trabajos de VPC
- Trabajar con configuraciones de seguridad en la consola de AWS Glue

Configuración de AWS Glue para utilizar configuraciones de seguridad

Siga estos pasos para configurar el entorno de AWS Glue para utilizar configuraciones de seguridad.

1. Cree o actualice las claves de AWS Key Management Service (AWS KMS) para conceder permisos de AWS KMS a los roles de IAM que se transfieren a los rastreadores y los trabajos de AWS Glue para cifrar registros de CloudWatch Logs. Para obtener más información, consulte

Cifrado de datos de registro en CloudWatch Logs con AWS KMS en la Guía del usuario de Amazon CloudWatch Logs.

En el siguiente ejemplo, *"role1"*, *"role2"* y *"role3"* son roles de IAM que se transfieren a los rastreadores y los trabajos.

```
{
       "Effect": "Allow",
       "Principal": { "Service": "logs.region.amazonaws.com",
       "AWS": [
                "role1",
                "role2",
                "role3"
             ] },
                    "Action": [
                           "kms:Encrypt*",
                           "kms:Decrypt*",
                           "kms:ReEncrypt*",
                           "kms:GenerateDataKey*",
                           "kms:Describe*"
                    ],
                    "Resource": "*"
}
```

La instrucción `Service`, que se muestra como `"Service": "logs.region.amazonaws.com"`, es necesaria si utiliza la clave para cifrar CloudWatch Logs.

2. Asegúrese de que la clave AWS KMS es `ENABLED` antes de utilizarla.

> **Note**
>
> Si utiliza Iceberg como marco de lago de datos, las tablas sw Iceberg tienen sus propios mecanismos para habilitar el cifrado del lado del servidor. Debe habilitar esta configuración además de las configuraciones de seguridad de AWS Glue. Para habilitar el cifrado del lado del servidor en las tablas de Iceberg, consulte las instrucciones de la documentación de Iceberg.

Creación de una ruta a AWS KMS para rastreadores y trabajos de VPC

Puede conectarse directamente a AWS KMS a través de un punto de enlace privado en su nube virtual privada (VPC) en lugar de conectarse a través de Internet. Cuando se utiliza un punto de enlace de la VPC, la comunicación entre la VPC y AWS KMS se realiza en su totalidad dentro de la red de AWS.

Puede crear un punto de enlace de la VPC de AWS KMS dentro de una VPC. Sin este paso, sus trabajos o rastreadores podrían generar un error con `kms timeout` en trabajos o con `internal service exception` en rastreadores. Para obtener instrucciones detalladas, consulte Conexión a AWS KMS a través de un punto de enlace de la VPC en la Guía para desarrolladores de AWS Key Management Service.

Cuando siga estas instrucciones, en la consola de VPC, debe hacer lo siguiente:

- Seleccione Enable Private DNS name (Habilitar nombre de DNS privado).
- Elija el Security group (Grupo de seguridad) (con regla de autorreferencia) que utilice para su trabajo o rastreador que accede a Java Database Connectivity (JDBC). Para obtener más información acerca de las conexiones de AWS Glue, consulte Conexión a datos.

Cuando agregue una configuración de seguridad a un rastreador o a un trabajo que accede a almacenes de datos de JDBC, AWS Glue debe tener una ruta al punto de enlace de la AWS KMS. Puede proporcionar la ruta con una gateway de conversión de las direcciones de red (NAT) o con un punto de enlace de las VPC de AWS KMS. Para crear una gateway NAT, consulte Gateways NAT en la Guía del usuario de Amazon VPC.

Trabajar con configuraciones de seguridad en la consola de AWS Glue

> **Warning**
>
> Las configuraciones de seguridad de AWS Glue actualmente no son compatibles con los trabajos de Ray.

Una configuración de seguridad en AWS Glue contiene las propiedades que se necesitan cuando se escriben datos cifrados. Las configuraciones de seguridad se crean en la consola de AWS Glue para proporcionar las propiedades de cifrado que utilizan los rastreadores, los flujos de trabajo y los puntos de enlace de desarrollo.

Para ver una lista de todas las configuraciones de seguridad que ha creado, abra la consola de AWS Glue en https://console.aws.amazon.com/glue/ y elija Security configurations (Configuraciones de seguridad) en el panel de navegación.

En la lista Security configurations (Configuraciones de seguridad), se muestran las siguientes propiedades para cada configuración:

Nombre

Nombre único proporcionado al crear la configuración. El nombre puede contener letras (A-Z), números (0-9), guiones (-) o guiones bajos (_) y tener un máximo de 255 caracteres.

Habilite el cifrado de Amazon S3

Si está activado, el modo de cifrado de Amazon Simple Storage Service (Amazon S3), como `SSE-KMS` o `SSE-S3`, está habilitado para el almacenamiento de metadatos en el catálogo de datos.

Habilite el cifrado de registros de Amazon CloudWatch

Si está activado, el modo de cifrado de Amazon S3, como `SSE-KMS`, se habilita al escribir registros en Amazon CloudWatch.

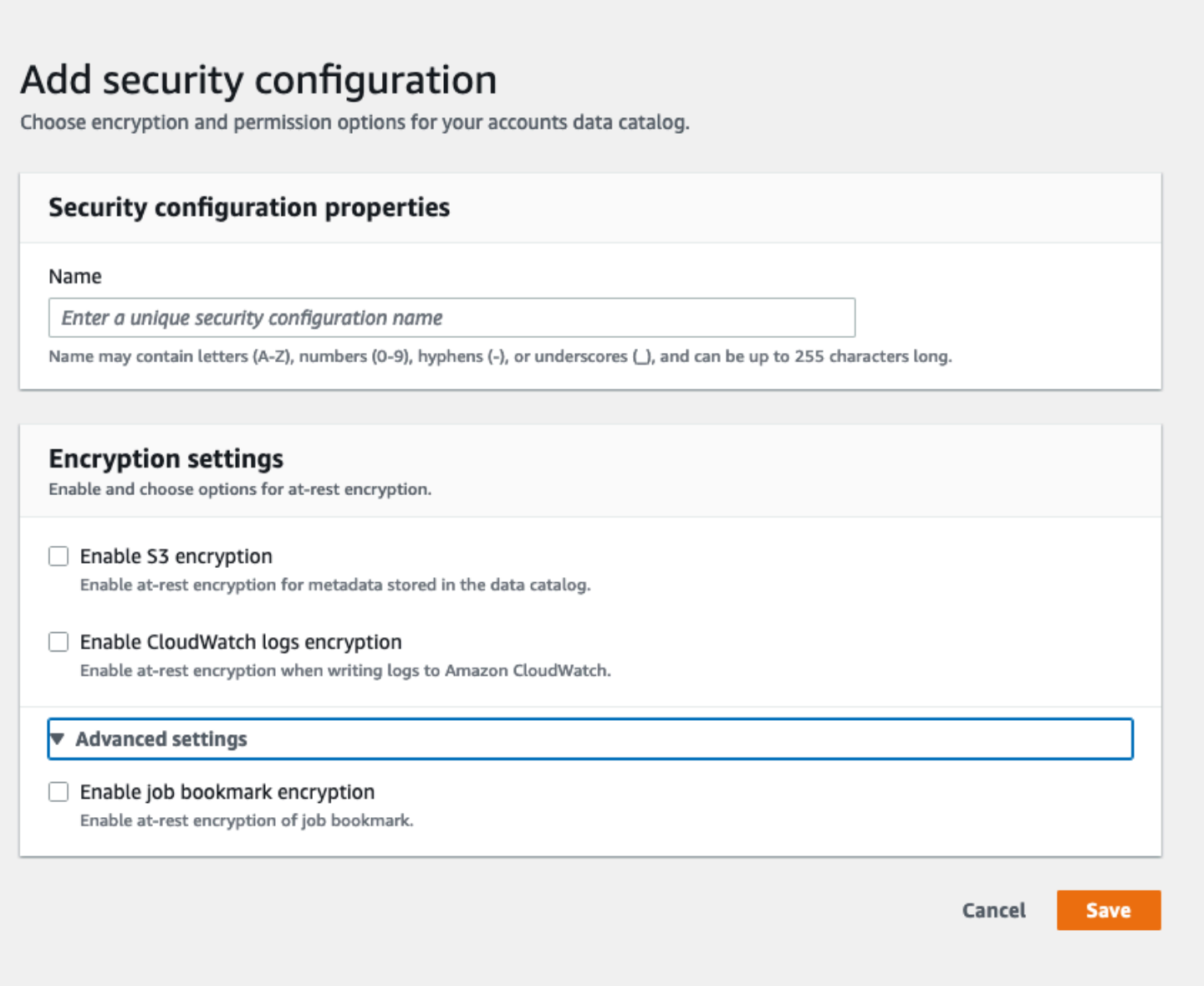
Configuración avanzada: habilitar el cifrado de marcadores de trabajo

Si está activado, el modo de cifrado de Amazon S3, como `CSE-KMS`, se habilita cuando se marcan los trabajos.

Puede agregar o eliminar configuraciones en la sección Security configurations (Configuraciones de seguridad) en la consola. Para consultar más detalles sobre una configuración seleccione el nombre de la configuración en la lista. Los detalles incluirán la información que definió al crear la configuración.

Agregado de una configuración de seguridad

Para agregar una configuración de seguridad a través de la consola de AWS Glue, en la página Security configurations (Configuraciones de seguridad), seleccione Add security configuration (Agregar configuración de seguridad).



Propiedades de configuración de seguridad

Introduzca un nombre de configuración de seguridad único. El nombre puede contener letras (A-Z), números (0-9), guiones (-) o guiones bajos (_) y tener un máximo de 255 caracteres.

Configuración de cifrado

Puede habilitar el cifrado en reposo para los metadatos almacenados en el Data Catalog de Amazon S3 y los registros en Amazon CloudWatch. Para configurar el cifrado de datos y metadatos con claves de AWS Key Management Service (AWS KMS) en la consola de AWS Glue, agregue una política al usuario de la consola. Esta política debe especificar los recursos permitidos como los

nombres de recurso de Amazon (ARN) de las claves que se utilizan para cifrar almacenes de datos de Amazon S3, tal y como se muestra en el siguiente ejemplo.

```
{
"Version": "2012-10-17",
  "Statement": {
  "Effect": "Allow",
  "Action": [
    "kms:GenerateDataKey",
    "kms:Decrypt",
    "kms:Encrypt"],
  "Resource": "arn:aws:kms:region:account-id:key/key-id"}
}
```

> **Important**
>
> Cuando una configuración de seguridad se asocia a un rastreador o trabajo, el rol de IAM que se transfiere debe tener permisos de AWS KMS. Para obtener más información, consulte Cifrado de datos escritos por AWS Glue.

Cuando se define una conexión, puede proporcionar valores para las siguientes propiedades:

Habilitar el cifrado de S3

Al escribir los datos de Amazon S3, puede utilizar el cifrado en el lado del servidor con claves administradas de Amazon S3 (SSE-S3) o el cifrado del lado del servidor con claves administradas de AWS KMS (SSE-KMS). Este campo es opcional. Para habilitar el acceso a Amazon S3, elija una clave AWS KMS, o elija Enter a key ARN (Ingresar un ARN de clave) y proporcione el ARN de la clave. Escriba el ARN con la forma `arn:aws:kms:region:account-id:key/key-id`. También puede proporcionar el ARN como un alias de clave, como `arn:aws:kms:region:account-id:alias/alias-name`.

Si habilita la interfaz de usuario de Spark para su trabajo, el archivo de registro de la interfaz de usuario de Spark cargado en Amazon S3 se aplicará con el mismo cifrado.

> **Important**
>
> AWS Glue solo soporta claves maestras de cliente (CMK) simétricas. La lista de AWS KMS key (Clave KMS) muestra únicamente claves simétricas. Sin embargo, si selecciona

Choose a AWS KMS key ARN (Elegir un ARN de clave KMS), la consola le permite introducir un ARN para cualquier tipo de clave. Asegúrese de introducir sólo ARN para claves simétricas.

Habilitar el cifrado de registros de CloudWatch

Se utiliza el cifrado del lado del servidor (SSE-KMS) para cifrar registros de CloudWatch Logs. Este campo es opcional. Para habilitarlo, elija una clave AWS KMS o elija Enter a key ARN (Ingresar un ARN de clave) y proporcione el ARN para la clave. Escriba el ARN con la forma `arn:aws:kms:`*`region`*`:`*`account-id`*`:key/`*`key-id`*. También puede proporcionar el ARN como un alias de clave, como `arn:aws:kms:`*`region`*`:`*`account-id`*`:alias/`*`alias-name`*.

Configuración avanzada: cifrado de marcadores de trabajo

El cifrado del lado del cliente (CSE-KMS) se utiliza para cifrar marcadores de trabajos. Este campo es opcional. Los datos del marcador se cifran antes de enviarlos a Amazon S3 para su almacenamiento. Para habilitarlo, elija una clave AWS KMS o elija Enter a key ARN (Ingresar un ARN de clave) y proporcione el ARN para la clave. Escriba el ARN con la forma `arn:aws:kms:`*`region`*`:`*`account-id`*`:key/`*`key-id`*. También puede proporcionar el ARN como un alias de clave, como `arn:aws:kms:`*`region`*`:`*`account-id`*`:alias/`*`alias-name`*.

Para obtener más información, consulte los siguientes temas en la Guía del usuario de Amazon Simple Storage Service:

- Para obtener más información acerca de `SSE-S3`, consulte Protección de los datos con el cifrado del lado del servidor con claves de cifrado administradas por Amazon S3 (SSE-S3).
- Para obtener más información sobre `SSE-KMS`, consulte Protección de los datos con el cifrado del lado del servidor con AWS KMS keys.
- Para obtener más información sobre `CSE-KMS`, consulte Cómo usar una clave KMS almacenada en AWS KMS.

Cifrado en tránsito

AWS ofrece cifrado de seguridad de la capa de transporte (TLS) para los datos en movimiento. Puede establecer la configuración de cifrado para rastreadores, trabajos de ETL y puntos de enlace

de desarrollo mediante configuraciones de seguridad en AWS Glue. Puede habilitar el cifrado AWS Glue Data Catalog a través de la configuración del Catálogo de datos.

A partir del 4 de septiembre de 2018, se soporta AWS KMS (traiga su propia clave y cifrado del lado del servidor) para ETL de AWS Glue y la AWS Glue Data Catalog.

Conformidad con FIPS

Si necesita módulos criptográficos validados FIPS 140-2 al acceder a AWS a través de una interfaz de la línea de comandos o una API, utilice un punto de conexión de FIPS. Para obtener más información sobre los puntos de conexión de FIPS disponibles, consulte Estándar de procesamiento de la información federal (FIPS) 140-2.

Administración de claves

Puede utilizar AWS Identity and Access Management (IAM) con AWS Glue para definir usuarios, recursos de AWS, grupos, roles y políticas detalladas en relación con el acceso, la denegación, etc.

Puede definir el acceso a los metadatos con ambas políticas basadas en recursos e identidades en función de las necesidades de su organización. Las políticas basadas en recursos enumeran las entidades principales a las que se le permite o deniega el acceso a sus recursos, lo que le permite configurar políticas como el acceso entre cuentas. Las políticas de identidades se asocian a los usuarios, grupos y roles dentro de IAM, de manera específica.

Para obtener un ejemplo paso a paso, consulte Restricción de acceso a AWS Glue Data Catalog con premisos de IAM a nivel de recursos y políticas con base en recursos en el blog de Big Data de AWS.

La parte de acceso detallado de la política se define en la cláusula `Resource`. Esta parte define tanto el objeto de AWS Glue Data Catalog en el que se puede realizar la acción como los objetos resultantes devueltos por esa operación.

Un punto de enlace de desarrollo es un entorno que puede utilizar para desarrollar y probar los scripts de AWS Glue. Puede agregar, eliminar o rotar la clave SSH de un punto de enlace de desarrollo.

A partir del 4 de septiembre de 2018, se soporta AWS KMS (traiga su propia clave y cifrado del lado del servidor) para ETL de AWS Glue y la AWS Glue Data Catalog.

Dependencia de AWS Glue en otros servicios de AWS

Para que un usuario pueda trabajar con la consola de AWS Glue, debe tener un conjunto mínimo de permisos que le permitan trabajar con los recursos de AWS Glue en su cuenta de AWS. Además de estos permisos de AWS Glue, la consola requiere permisos de los servicios a continuación:

- Permisos de Amazon CloudWatch Logs para mostrar registros.
- Permisos de AWS Identity and Access Management (IAM) para enumerar y transferir roles.
- Permisos AWS CloudFormation para trabajar con pilas.
- Permisos de Amazon Elastic Compute Cloud (Amazon EC2) para enumerar las nubes privadas virtuales (VPC), subredes, grupos de seguridad, instancias y otros objetos (para configurar elementos de Amazon EC2 como VPC cuando se ejecutan trabajos y rastreadores, y se crean puntos de enlace de desarrollo).
- Permisos de Amazon Simple Storage Service (Amazon S3) para enumerar buckets y objetos, y recuperar y guardar scripts.
- Permisos necesarios de Amazon Redshift para trabajar con clústeres.
- Permisos de Amazon Relational Database Service (Amazon RDS) para enumerar instancias.

Puntos de conexión de desarrollo

Un punto de enlace de desarrollo es un entorno que puede utilizar para desarrollar y probar los scripts de AWS Glue. Puede usar AWS Glue para crear, editar y eliminar puntos de enlace de desarrollo. Puede enumerar todos los puntos conexión de desarrollo creados. Puede agregar, eliminar o rotar la clave SSH de un punto de enlace de desarrollo. También puede crear blocs de notas que usen el punto de enlace de desarrollo.

Puede proporcionar valores de configuración para aprovisionar los entornos de desarrollo. Estos valores indican a AWS Glue cómo configurar la red para que pueda obtener acceso a su punto de enlace de desarrollo de forma segura y que su punto de enlace pueda obtener acceso a sus almacenes de datos. A continuación, puede crear un bloc de notas que se conecte al punto de enlace de desarrollo. Puede utilizar el bloc de notas para crear y probar su script de ETL.

Use un rol de AWS Identity and Access Management (IAM) con permisos similares al rol de IAM que utiliza para ejecutar trabajos de ETL de AWS Glue. Utilice una nube privada virtual (VPC), una subred, y un grupo de seguridad para crear un punto de enlace de desarrollo que se pueda conectar

a sus recursos de datos de forma segura. Puede generar un par de claves SSH para conectarse al entorno de desarrollo mediante SSH.

Puede crear puntos de enlace de desarrollo para datos de Amazon S3 y dentro de una VPC que puede utilizar para acceder a conjuntos de datos a través de JDBC.

Puede instalar un cliente de cuaderno de Jupyter en su equipo local y utilizarlo para depurar y probar scripts ETL en un punto de conexión de desarrollo. O bien, puede usar un cuaderno de Sagemaker para crear scripts de ETL en JupyterLab en AWS. Consulte Uso de un cuaderno de SageMaker con su punto de conexión de desarrollo.

AWS Glue etiqueta las instancias de Amazon EC2 con un nombre prefijado con `aws-glue-dev-endpoint`.

Puede configurar un servidor de cuadernos en un punto de conexión de desarrollo para ejecutar PySpark con extensiones de AWS Glue.

Gestión de identidad y acceso para AWS Glue

AWS Identity and Access Management (IAM) es una herramienta Servicio de AWS que ayuda al administrador a controlar de forma segura el acceso a los AWS recursos. Los administradores de IAM controlan quién puede autenticarse (iniciar sesión) y quién puede autorizarse (tener permisos) para usar los recursos de AWS Glue. Puedes usar IAM sin coste adicional. Servicio de AWS

> **Note**
>
> Puede conceder acceso a sus datos en el catálogo de datos de AWS Glue mediante AWS Glue métodos o AWS Lake Formation concesiones. Puede usar políticas AWS Identity and Access Management (IAM) para establecer un control de acceso detallado con métodos. AWS Glue Lake Formation utiliza un modelo de permisos `GRANT/REVOKE` más simple que es similar a los comandos `GRANT/REVOKE` en un sistema de base de datos relacional.
> Esta sección incluye información sobre cómo usar los métodos de AWS Glue. Para obtener más información sobre cómo utilizar las concesiones de Lake Formation, consulte Concesión de permisos de Lake Formation en la Guía para desarrolladores de AWS Lake Formation .

Temas

- Público
- Autenticación con identidades
- Administración de acceso mediante políticas
- Cómo funciona AWS Glue con IAM
- Configuración de permisos de IAM para AWS Glue
- Ejemplos de políticas de control de acceso de AWS Glue
- AWS políticas gestionadas para AWS Glue
- Especificación de ARN del recurso de AWS Glue
- Concesión de acceso entre cuentas
- Solución de problemas de identidad y acceso a AWS Glue

Público

La forma de usar AWS Identity and Access Management (IAM) varía según el trabajo que realices en AWS Glue.

Usuario del servicio: si utilizas el servicio AWS Glue para realizar tu trabajo, el administrador te proporcionará las credenciales y los permisos que necesitas. A medida que utilices más funciones de AWS Glue para realizar tu trabajo, es posible que necesites permisos adicionales. Entender cómo se administra el acceso puede ayudarlo a solicitar los permisos correctos al administrador. Si no puede acceder a una característica en AWS Glue, consulte Solución de problemas de identidad y acceso a AWS Glue.

Administrador de servicios: si está a cargo de los recursos de AWS Glue en su empresa, probablemente tenga acceso completo a AWS Glue. Es tu trabajo determinar a qué funciones y recursos de AWS Glue deben acceder los usuarios del servicio. Luego, debe enviar solicitudes a su administrador de IAM para cambiar los permisos de los usuarios de su servicio. Revise la información de esta página para conocer los conceptos básicos de IAM. Para obtener más información sobre cómo su empresa puede utilizar IAM con AWS Glue, consulteCómo funciona AWS Glue con IAM.

Administrador de IAM: si eres administrador de IAM, quizá te interese obtener más información sobre cómo puedes redactar políticas para gestionar el acceso a Glue AWS . Para ver ejemplos de políticas basadas en la identidad de AWS Glue que puede utilizar en IAM, consulte. Ejemplos de políticas basadas en identidades para AWS Glue

Autenticación con identidades

La autenticación es la forma en que inicias sesión AWS con tus credenciales de identidad. Debe estar autenticado (con quien haya iniciado sesión AWS) como usuario de IAM o asumiendo una función de IAM. Usuario raíz de la cuenta de AWS

Puede iniciar sesión AWS como una identidad federada mediante las credenciales proporcionadas a través de una fuente de identidad. AWS IAM Identity Center Los usuarios (Centro de identidades de IAM), la autenticación de inicio de sesión único de su empresa y sus credenciales de Google o Facebook son ejemplos de identidades federadas. Al iniciar sesión como una identidad federada, su administrador habrá configurado previamente la federación de identidades mediante roles de IAM. Cuando accedes AWS mediante la federación, estás asumiendo un rol de forma indirecta.

Según el tipo de usuario que sea, puede iniciar sesión en el portal AWS Management Console o en el de AWS acceso. Para obtener más información sobre cómo iniciar sesión AWS, consulte Cómo iniciar sesión Cuenta de AWS en su Guía del AWS Sign-In usuario.

Si accede AWS mediante programación, AWS proporciona un kit de desarrollo de software (SDK) y una interfaz de línea de comandos (CLI) para firmar criptográficamente sus solicitudes con sus credenciales. Si no utilizas AWS herramientas, debes firmar las solicitudes tú mismo. Para obtener más información sobre cómo usar el método recomendado para firmar las solicitudes usted mismo, consulte Firmar las solicitudes de la AWS API en la Guía del usuario de IAM.

Independientemente del método de autenticación que use, es posible que deba proporcionar información de seguridad adicional. Por ejemplo, le AWS recomienda que utilice la autenticación multifactor (MFA) para aumentar la seguridad de su cuenta. Para obtener más información, consulte Autenticación multifactor en la Guía del usuario de AWS IAM Identity Center y Uso de la autenticación multifactor (MFA) en AWSen la Guía del usuario de IAM.

Cuenta de AWS usuario root

Al crear una Cuenta de AWS, comienza con una identidad de inicio de sesión que tiene acceso completo a todos Servicios de AWS los recursos de la cuenta. Esta identidad se denomina usuario Cuenta de AWS raíz y se accede a ella iniciando sesión con la dirección de correo electrónico y la contraseña que utilizaste para crear la cuenta. Recomendamos encarecidamente que no utilice el usuario raíz para sus tareas diarias. Proteja las credenciales del usuario raíz y utilícelas solo para las tareas que solo el usuario raíz pueda realizar. Para ver la lista completa de las tareas que requieren que inicie sesión como usuario raíz, consulte Tareas que requieren credenciales de usuario raíz en la Guía del usuario de IAM.

Identidad federada

Como práctica recomendada, exija a los usuarios humanos, incluidos los que requieren acceso de administrador, que utilicen la federación con un proveedor de identidades para acceder Servicios de AWS mediante credenciales temporales.

Una identidad federada es un usuario del directorio de usuarios de su empresa, un proveedor de identidades web AWS Directory Service, el directorio del Centro de Identidad o cualquier usuario al que acceda Servicios de AWS mediante las credenciales proporcionadas a través de una fuente de identidad. Cuando las identidades federadas acceden Cuentas de AWS, asumen funciones y las funciones proporcionan credenciales temporales.

Para una administración de acceso centralizada, le recomendamos que utilice AWS IAM Identity Center. Puede crear usuarios y grupos en el Centro de identidades de IAM o puede conectarse y sincronizarse con un conjunto de usuarios y grupos de su propia fuente de identidad para usarlos en todas sus Cuentas de AWS aplicaciones. Para obtener más información, consulte ¿Qué es el Centro de identidades de IAM? en la Guía del usuario de AWS IAM Identity Center .

Usuarios y grupos de IAM

Un usuario de IAM es una identidad propia Cuenta de AWS que tiene permisos específicos para una sola persona o aplicación. Siempre que sea posible, recomendamos emplear credenciales temporales, en lugar de crear usuarios de IAM que tengan credenciales de larga duración como contraseñas y claves de acceso. No obstante, si tiene casos de uso específicos que requieran credenciales de larga duración con usuarios de IAM, recomendamos rotar las claves de acceso. Para más información, consulte Rotar las claves de acceso periódicamente para casos de uso que requieran credenciales de larga duración en la Guía del usuario de IAM.

Un grupo de IAM es una identidad que especifica un conjunto de usuarios de IAM. No puede iniciar sesión como grupo. Puede usar los grupos para especificar permisos para varios usuarios a la vez. Los grupos facilitan la administración de los permisos de grandes conjuntos de usuarios. Por ejemplo, podría tener un grupo cuyo nombre fuese IAMAdmins y conceder permisos a dicho grupo para administrar los recursos de IAM.

Los usuarios son diferentes de los roles. Un usuario se asocia exclusivamente a una persona o aplicación, pero la intención es que cualquier usuario pueda asumir un rol que necesite. Los usuarios tienen credenciales permanentes a largo plazo y los roles proporcionan credenciales temporales. Para más información, consulte Cuándo crear un usuario de IAM (en lugar de un rol) en la Guía del usuario de IAM.

Roles de IAM

Un rol de IAM es una identidad dentro de usted Cuenta de AWS que tiene permisos específicos. Es similar a un usuario de IAM, pero no está asociado a una determinada persona. Puede asumir temporalmente una función de IAM en el AWS Management Console cambiando de función. Puede asumir un rol llamando a una operación de AWS API AWS CLI o utilizando una URL personalizada. Para más información sobre los métodos para el uso de roles, consulte Uso de roles de IAM en la Guía del usuario de IAM.

Los roles de IAM con credenciales temporales son útiles en las siguientes situaciones:

- Acceso de usuario federado: para asignar permisos a una identidad federada, puede crear un rol y definir sus permisos. Cuando se autentica una identidad federada, se asocia la identidad al rol y se le conceden los permisos define el rol. Para obtener información acerca de roles para federación, consulte Creación de un rol para un proveedor de identidades de terceros en la Guía del usuario de IAM. Si utiliza IAM Identity Center, debe configurar un conjunto de permisos. IAM Identity Center correlaciona el conjunto de permisos con un rol en IAM para controlar a qué pueden acceder las identidades después de autenticarse. Para obtener información acerca de los conjuntos de permisos, consulte Conjuntos de permisos en la Guía del usuario de AWS IAM Identity Center .
- Permisos de usuario de IAM temporales: un usuario de IAM puede asumir un rol de IAM para recibir temporalmente permisos distintos que le permitan realizar una tarea concreta.
- Acceso entre cuentas: puede utilizar un rol de IAM para permitir que alguien (una entidad principal de confianza) de otra cuenta acceda a los recursos de la cuenta. Los roles son la forma principal de conceder acceso entre cuentas. Sin embargo, con algunas Servicios de AWS, puedes adjuntar una política directamente a un recurso (en lugar de usar un rol como proxy). Para obtener información acerca de la diferencia entre los roles y las políticas basadas en recursos para el acceso entre cuentas, consulte Acceso a recursos entre cuentas en IAM en la Guía del usuario de IAM.
- Acceso entre servicios: algunos Servicios de AWS utilizan funciones en otros Servicios de AWS. Por ejemplo, cuando realiza una llamada en un servicio, es común que ese servicio ejecute aplicaciones en Amazon EC2 o almacene objetos en Amazon S3. Es posible que un servicio haga esto usando los permisos de la entidad principal, usando un rol de servicio o usando un rol vinculado al servicio.
  - Sesiones de acceso directo (FAS): cuando utilizas un usuario o un rol de IAM para realizar acciones en ellas AWS, se te considera director. Cuando utiliza algunos servicios, es posible que realice una acción que desencadene otra acción en un servicio diferente. El FAS utiliza los

permisos del principal que llama Servicio de AWS y los solicita Servicio de AWS para realizar solicitudes a los servicios descendentes. Las solicitudes de FAS solo se realizan cuando un servicio recibe una solicitud que requiere interacciones con otros Servicios de AWS recursos para completarse. En este caso, debe tener permisos para realizar ambas acciones. Para obtener información sobre las políticas a la hora de realizar solicitudes de FAS, consulte Reenviar sesiones de acceso.

- Rol de servicio: un rol de servicio es un rol de IAM que adopta un servicio para realizar acciones en su nombre. Un administrador de IAM puede crear, modificar y eliminar un rol de servicio desde IAM. Para obtener más información, consulte Creación de un rol para delegar permisos a un Servicio de AWS en la Guía del usuario de IAM.
- Función vinculada al servicio: una función vinculada a un servicio es un tipo de función de servicio que está vinculada a un. Servicio de AWS El servicio puede asumir el rol para realizar una acción en su nombre. Los roles vinculados al servicio aparecen en usted Cuenta de AWS y son propiedad del servicio. Un administrador de IAM puede ver, pero no editar, los permisos de los roles vinculados a servicios.

- Aplicaciones que se ejecutan en Amazon EC2: puede usar un rol de IAM para administrar las credenciales temporales de las aplicaciones que se ejecutan en una instancia EC2 y realizan AWS CLI solicitudes a la API. AWS Es preferible hacerlo de este modo a almacenar claves de acceso en la instancia de EC2. Para asignar una AWS función a una instancia EC2 y ponerla a disposición de todas sus aplicaciones, debe crear un perfil de instancia adjunto a la instancia. Un perfil de instancia contiene el rol y permite a los programas que se ejecutan en la instancia de EC2 obtener credenciales temporales. Para más información, consulte Uso de un rol de IAM para conceder permisos a aplicaciones que se ejecutan en instancias Amazon EC2 en la Guía del usuario de IAM.

Para obtener información sobre el uso de los roles de IAM, consulte Cuándo crear un rol de IAM (en lugar de un usuario) en la Guía del usuario de IAM.

Administración de acceso mediante políticas

El acceso se controla AWS creando políticas y adjuntándolas a AWS identidades o recursos. Una política es un objeto AWS que, cuando se asocia a una identidad o un recurso, define sus permisos. AWS evalúa estas políticas cuando un director (usuario, usuario raíz o sesión de rol) realiza una solicitud. Los permisos en las políticas determinan si la solicitud se permite o se deniega. La mayoría de las políticas se almacenan AWS como documentos JSON. Para obtener más información sobre

la estructura y el contenido de los documentos de política JSON, consulte Información general de políticas JSON en la Guía del usuario de IAM.

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

De forma predeterminada, los usuarios y los roles no tienen permisos. Un administrador de IAM puede crear políticas de IAM para conceder permisos a los usuarios para realizar acciones en los recursos que necesitan. A continuación, el administrador puede añadir las políticas de IAM a roles y los usuarios pueden asumirlos.

Las políticas de IAM definen permisos para una acción independientemente del método que se utilice para realizar la operación. Por ejemplo, suponga que dispone de una política que permite la acción `iam:GetRole`. Un usuario con esa política puede obtener información sobre el rol de la API AWS Management Console AWS CLI, la o la AWS API.

Políticas basadas en identidades

Las políticas basadas en identidad son documentos de políticas de permisos JSON que puede asociar a una identidad, como un usuario de IAM, un grupo de usuarios o un rol. Estas políticas controlan qué acciones pueden realizar los usuarios y los roles, en qué recursos y en qué condiciones. Para obtener más información sobre cómo crear una política basada en identidad, consulte Creación de políticas de IAM en la Guía del usuario de IAM.

Las políticas basadas en identidades pueden clasificarse además como políticas insertadas o políticas administradas. Las políticas insertadas se integran directamente en un único usuario, grupo o rol. Las políticas administradas son políticas independientes que puede adjuntar a varios usuarios, grupos y roles de su Cuenta de AWS empresa. Las políticas administradas incluyen políticas AWS administradas y políticas administradas por el cliente. Para más información sobre cómo elegir una política administrada o una política insertada, consulte Elegir entre políticas administradas y políticas insertadas en la Guía del usuario de IAM.

Políticas basadas en recursos

Las políticas basadas en recursos son documentos de política JSON que se asocian a un recurso. Ejemplos de políticas basadas en recursos son las políticas de confianza de roles de IAM y las políticas de bucket de Amazon S3. En los servicios que admiten políticas basadas en recursos, los administradores de servicios pueden utilizarlos para controlar el acceso a un recurso específico. Para el recurso al que se asocia la política, la política define qué acciones puede realizar una entidad

principal especificada en ese recurso y en qué condiciones. Debe especificar una entidad principal en una política en función de recursos. Los principales pueden incluir cuentas, usuarios, roles, usuarios federados o. Servicios de AWS

Las políticas basadas en recursos son políticas insertadas que se encuentran en ese servicio. No puedes usar políticas AWS gestionadas de IAM en una política basada en recursos.

Listas de control de acceso (ACL)

Las listas de control de acceso (ACL) controlan qué entidades principales (miembros de cuentas, usuarios o roles) tienen permisos para acceder a un recurso. Las ACL son similares a las políticas basadas en recursos, aunque no utilizan el formato de documento de políticas JSON.

Amazon S3 y Amazon VPC son ejemplos de servicios que admiten las ACL. AWS WAF Para obtener más información sobre las ACL, consulte Información general de Lista de control de acceso (ACL) en la Guía para desarrolladores de Amazon Simple Storage Service.

Otros tipos de políticas

AWS admite tipos de políticas adicionales y menos comunes. Estos tipos de políticas pueden establecer el máximo de permisos que los tipos de políticas más frecuentes le conceden.

- Límites de permisos: un límite de permisos es una característica avanzada que le permite establecer los permisos máximos que una política basada en identidad puede conceder a una entidad de IAM (usuario o rol de IAM). Puede establecer un límite de permisos para una entidad. Los permisos resultantes son la intersección de las políticas basadas en la identidad de la entidad y los límites de permisos. Las políticas basadas en recursos que especifiquen el usuario o rol en el campo `Principal` no estarán restringidas por el límite de permisos. Una denegación explícita en cualquiera de estas políticas anulará el permiso. Para obtener más información sobre los límites de los permisos, consulte Límites de permisos para las entidades de IAM en la Guía del usuario de IAM.
- Políticas de control de servicios (SCP): las SCP son políticas de JSON que especifican los permisos máximos para una organización o unidad organizativa (OU). AWS Organizations AWS Organizations es un servicio para agrupar y gestionar de forma centralizada varios de los Cuentas de AWS que son propiedad de su empresa. Si habilita todas las características en una organización, entonces podrá aplicar políticas de control de servicio (SCP) a una o a todas sus cuentas. El SCP limita los permisos de las entidades en las cuentas de los miembros, incluidas las de cada una. Usuario raíz de la cuenta de AWS Para obtener más información acerca de

Organizations y las SCP, consulte Funcionamiento de las SCP en la Guía del usuario de AWS Organizations .

- Políticas de sesión: las políticas de sesión son políticas avanzadas que se pasan como parámetro cuando se crea una sesión temporal mediante programación para un rol o un usuario federado. Los permisos de la sesión resultantes son la intersección de las políticas basadas en identidades del rol y las políticas de la sesión. Los permisos también pueden proceder de una política en función de recursos. Una denegación explícita en cualquiera de estas políticas anulará el permiso. Para más información, consulte Políticas de sesión en la Guía del usuario de IAM.

Varios tipos de políticas

Cuando se aplican varios tipos de políticas a una solicitud, los permisos resultantes son más complicados de entender. Para saber cómo AWS determinar si se debe permitir una solicitud cuando se trata de varios tipos de políticas, consulte la lógica de evaluación de políticas en la Guía del usuario de IAM.

Cómo funciona AWS Glue con IAM

Antes de utilizar IAM para gestionar el acceso a AWS Glue, infórmese sobre las funciones de IAM disponibles para su uso con Glue AWS .

Funciones de IAM que puedes usar con Glue AWS

Característica de IAM	AWS Soporte de pegamento
Políticas basadas en identidades	Sí
Políticas basadas en recursos	Parcial
Acciones de políticas	Sí
Recursos de políticas	Sí
Claves de condición de política (específicas del servicio)	Sí
ACL	No

Característica de IAM	AWS Soporte de pegamento
ABAC (etiquetas en políticas)	Parcial
Credenciales temporales	Sí
Permisos de entidades principales	No
Roles de servicio	Sí
Roles vinculados al servicio	No

Para obtener una visión general de cómo funcionan AWS Glue y otros AWS servicios con la mayoría de las funciones de IAM, consulte AWS los servicios que funcionan con IAM en la Guía del usuario de IAM.

Políticas basadas en la identidad para Glue AWS

Compatibilidad con las políticas basadas en identidad	Sí

Las políticas basadas en identidad son documentos de políticas de permisos JSON que puede asociar a una identidad, como un usuario de IAM, un grupo de usuarios o un rol. Estas políticas controlan qué acciones pueden realizar los usuarios y los roles, en qué recursos y en qué condiciones. Para obtener más información sobre cómo crear una política basada en identidad, consulte Creación de políticas de IAM en la Guía del usuario de IAM.

Con las políticas basadas en identidades de IAM, puede especificar las acciones y los recursos permitidos o denegados, así como las condiciones en las que se permiten o deniegan las acciones. No es posible especificar la entidad principal en una política basada en identidad porque se aplica al usuario o rol al que está adjunto. Para más información sobre los elementos que puede utilizar en una política de JSON, consulte Referencia de los elementos de las políticas de JSON de IAM en la Guía del usuario de IAM.

AWS Glue admite políticas basadas en identidad (políticas de IAM) para todas las operaciones de AWS Glue. Al asociar una política, puede conceder permisos para crear o modificar un recurso de AWS Glue, como una tabla en AWS Glue Data Catalog, o para acceder a él.

Ejemplos de políticas basadas en la identidad para Glue AWS

Para ver ejemplos de políticas de AWS Glue basadas en la identidad, consulte. Ejemplos de políticas basadas en identidades para AWS Glue

Políticas basadas en recursos dentro de Glue AWS

Compatibilidad con las políticas basadas en recursos	Parcial

Las políticas basadas en recursos son documentos de política JSON que se asocian a un recurso. Ejemplos de políticas basadas en recursos son las políticas de confianza de roles de IAM y las políticas de bucket de Amazon S3. En los servicios que admiten políticas basadas en recursos, los administradores de servicios pueden utilizarlos para controlar el acceso a un recurso específico. Para el recurso al que se asocia la política, la política define qué acciones puede realizar una entidad principal especificada en ese recurso y en qué condiciones. Debe especificar una entidad principal en una política en función de recursos. Los principales pueden incluir cuentas, usuarios, roles, usuarios federados o. Servicios de AWS

Para habilitar el acceso entre cuentas, puede especificar toda una cuenta o entidades de IAM de otra cuenta como la entidad principal de una política en función de recursos. Añadir a una política en función de recursos una entidad principal entre cuentas es solo una parte del establecimiento de una relación de confianza. Cuando el principal y el recurso son diferentes Cuentas de AWS, el administrador de IAM de la cuenta de confianza también debe conceder a la entidad principal (usuario o rol) permiso para acceder al recurso. Para conceder el permiso, adjunte la entidad a una política basada en identidad. Sin embargo, si la política en función de recursos concede el acceso a una entidad principal de la misma cuenta, no es necesaria una política basada en identidad adicional. Para obtener más información, consulte Acceso a recursos entre cuentas en IAM en la Guía del usuario de IAM.

> **Note**
>
> Solo puede usar una política de recursos de AWS Glue para administrar los permisos de los recursos del Catálogo de datos. No se puede adjuntar a otros recursos de AWS Glue como trabajos, disparadores, puntos de enlace de desarrollo, rastreadores o clasificadores. Solo se permite una política de recursos por catálogo y su tamaño se limita a 10 KB.

En AWS Glue, se adjunta una política de recursos a un catálogo, que es un contenedor virtual para todos los tipos de recursos del catálogo de datos mencionados anteriormente. Cada AWS cuenta es propietaria de un único catálogo en una AWS región cuyo identificador de catálogo es el mismo que el identificador de la AWS cuenta. No puede eliminar ni modificar un catálogo.

Una política de recursos se evalúa para todas las llamadas a la API en el catálogo donde el principal del intermediario se incluye en el bloque `"Principal"` del documento de política.

Para ver ejemplos de políticas basadas en recursos de AWS Glue, consulte. Ejemplos de políticas basadas en recursos para AWS Glue

Acciones políticas para AWS Glue

Admite acciones de política	Sí

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

El elemento `Action` de una política JSON describe las acciones que puede utilizar para conceder o denegar el acceso en una política. Las acciones políticas suelen tener el mismo nombre que la operación de AWS API asociada. Hay algunas excepciones, como acciones de solo permiso que no tienen una operación de API coincidente. También hay algunas operaciones que requieren varias acciones en una política. Estas acciones adicionales se denominan acciones dependientes.

Incluya acciones en una política para conceder permisos y así llevar a cabo la operación asociada.

Para ver una lista de las acciones de AWS Glue, consulte Acciones definidas por AWS Glue en la Referencia de autorización de servicio.

Las acciones políticas en AWS Glue usan el siguiente prefijo antes de la acción:

```
glue
```

Para especificar varias acciones en una única instrucción, sepárelas con comas.

```
"Action": [
      "glue:action1",
```

```
      "glue:action2"
         ]
```

Puede utilizar caracteres comodín (*) para especificar varias acciones . Por ejemplo, para especificar todas las acciones que comiencen con la palabra `Get`, incluya la siguiente acción:

```
"Action": "glue:Get*"
```

Para ver ejemplos de políticas, consulte Ejemplos de políticas de control de acceso de AWS Glue.

Recursos de políticas para AWS Glue

Admite recursos de políticas	Sí

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

El elemento `Resource` de la política JSON especifica el objeto u objetos a los que se aplica la acción. Las instrucciones deben contener un elemento `Resource` o `NotResource`. Como práctica recomendada, especifique un recurso utilizando el Nombre de recurso de Amazon (ARN). Puede hacerlo para acciones que admitan un tipo de recurso específico, conocido como permisos de nivel de recurso.

Para las acciones que no admiten permisos de nivel de recurso, como las operaciones de descripción, utilice un carácter comodín (*) para indicar que la instrucción se aplica a todos los recursos.

```
"Resource": "*"
```

Para obtener más información sobre cómo controlar el acceso a los recursos de AWS Glue mediante ARN, consulteEspecificación de ARN del recurso de AWS Glue.

Para ver una lista de los tipos de recursos de AWS Glue y sus ARN, consulte Recursos definidos por AWS Glue en la referencia de autorización de servicio. Para saber qué acciones puedes usar para especificar el ARN de cada recurso, consulta Acciones definidas por Glue AWS.

Claves de condiciones de la política para AWS Glue

Admite claves de condición de políticas específicas del servicio	Sí

Los administradores pueden usar las políticas de AWS JSON para especificar quién tiene acceso a qué. Es decir, qué entidad principal puede realizar acciones en qué recursos y en qué condiciones.

El elemento `Condition` (o bloque de `Condition`) permite especificar condiciones en las que entra en vigor una instrucción. El elemento `Condition` es opcional. Puede crear expresiones condicionales que utilicen operadores de condición, tales como igual o menor que, para que la condición de la política coincida con los valores de la solicitud.

Si especifica varios elementos de `Condition` en una instrucción o varias claves en un único elemento de `Condition`, AWS las evalúa mediante una operación `AND` lógica. Si especifica varios valores para una única clave de condición, AWS evalúa la condición mediante una `OR` operación lógica. Se deben cumplir todas las condiciones antes de que se concedan los permisos de la instrucción.

También puede utilizar variables de marcador de posición al especificar condiciones. Por ejemplo, puede conceder un permiso de usuario de IAM para acceder a un recurso solo si está etiquetado con su nombre de usuario de IAM. Para más información, consulte Elementos de la política de IAM: variables y etiquetas en la Guía del usuario de IAM.

AWS admite claves de condición globales y claves de condición específicas del servicio. Para ver todas las claves de condición AWS globales, consulte las claves de contexto de condición AWS globales en la Guía del usuario de IAM.

Para ver una lista de las claves de estado de AWS Glue, consulta las claves de estado de AWS Glue en la referencia de autorización de servicio. Para saber con qué acciones y recursos puedes usar una clave de condición, consulta Acciones definidas por AWS Glue.

Para ver ejemplos de políticas, consulte Controlar la configuración mediante claves de condición o claves de contexto.

ACL en Glue AWS

Admite las ACL	No

Las listas de control de acceso (ACL) controlan qué entidades principales (miembros de cuentas, usuarios o roles) tienen permisos para acceder a un recurso. Las ACL son similares a las políticas basadas en recursos, aunque no utilizan el formato de documento de políticas JSON.

ABAC con Glue AWS

Admite ABAC (etiquetas en las políticas)	Parcial

El control de acceso basado en atributos (ABAC) es una estrategia de autorización que define permisos en función de atributos. En AWS, estos atributos se denominan etiquetas. Puede adjuntar etiquetas a las entidades de IAM (usuarios o roles) y a muchos AWS recursos. El etiquetado de entidades y recursos es el primer paso de ABAC. A continuación, designa las políticas de ABAC para permitir operaciones cuando la etiqueta de la entidad principal coincida con la etiqueta del recurso al que se intenta acceder.

ABAC es útil en entornos que crecen con rapidez y ayuda en situaciones en las que la administración de las políticas resulta engorrosa.

Para controlar el acceso en función de etiquetas, debe proporcionar información de las etiquetas en el elemento de condición de una política utilizando las claves de condición `aws:ResourceTag/key-name`, `aws:RequestTag/key-name` o `aws:TagKeys`.

Si un servicio admite las tres claves de condición para cada tipo de recurso, el valor es Sí para el servicio. Si un servicio admite las tres claves de condición solo para algunos tipos de recursos, el valor es Parcial.

Para obtener más información sobre ABAC, consulte ¿Qué es ABAC? en la Guía del usuario de IAM. Para ver un tutorial con los pasos para configurar ABAC, consulte Uso del control de acceso basado en atributos (ABAC) en la Guía del usuario de IAM.

> **Important**
>
> Las claves de contexto de condición se aplican únicamente a las acciones de API de AWS Glue que se realizan en los rastreadores, los trabajos, los desencadenadores y los puntos de conexión de desarrollo. Para obtener más información sobre las operaciones de la API que se ven afectadas, consulta Claves de condición de AWS Glue.
> Las operaciones de API del Catálogo de datos de AWS Glue no admiten actualmente las claves de contexto de condiciones globales `aws:referer` y `aws:UserAgent`.

Para consultar un ejemplo de política basada en la identidad para limitar el acceso a un recurso en función de las etiquetas de ese recurso, consulte Conceder acceso mediante etiquetas.

Uso de credenciales temporales con AWS Glue

Compatible con el uso de credenciales temporales	Sí

Algunas Servicios de AWS no funcionan cuando inicias sesión con credenciales temporales. Para obtener información adicional, incluida la información sobre cuáles Servicios de AWS funcionan con credenciales temporales, consulta Cómo Servicios de AWS funcionan con IAM en la Guía del usuario de IAM.

Utiliza credenciales temporales si inicia sesión en ellas AWS Management Console mediante cualquier método excepto un nombre de usuario y una contraseña. Por ejemplo, cuando accedes AWS mediante el enlace de inicio de sesión único (SSO) de tu empresa, ese proceso crea automáticamente credenciales temporales. También crea credenciales temporales de forma automática cuando inicia sesión en la consola como usuario y luego cambia de rol. Para más información sobre el cambio de roles, consulte Cambio a un rol (consola) en la Guía del usuario de IAM.

Puedes crear credenciales temporales manualmente mediante la AWS CLI API o. AWS A continuación, puede utilizar esas credenciales temporales para acceder AWS. AWS recomienda generar credenciales temporales de forma dinámica en lugar de utilizar claves de acceso a largo plazo. Para más información, consulte Credenciales de seguridad temporales en IAM.

Permisos principales de servicios cruzados para Glue AWS

Admite sesiones de acceso directo (FAS)	No

Cuando utilizas un usuario o un rol de IAM para realizar acciones en él AWS, se te considera principal. Cuando utiliza algunos servicios, es posible que realice una acción que desencadene otra acción en un servicio diferente. FAS utiliza los permisos del principal que llama y los que solicita Servicio de AWS para realizar solicitudes a los servicios descendentes. Servicio de AWS Las solicitudes de FAS solo se realizan cuando un servicio recibe una solicitud que requiere interacciones con otros Servicios de AWS recursos para completarse. En este caso, debe tener permisos para

realizar ambas acciones. Para obtener información detallada sobre las políticas al momento de realizar solicitudes de FAS, consulte Reenviar las sesiones de acceso.

Roles de servicio para AWS Glue

Compatible con roles de servicio	Sí

Un rol de servicio es un rol de IAM que asume un servicio para realizar acciones en su nombre. Un administrador de IAM puede crear, modificar y eliminar un rol de servicio desde IAM. Para obtener más información, consulte Creación de un rol para delegar permisos a un Servicio de AWS en la Guía del usuario de IAM.

> **Warning**
>
> Cambiar los permisos de un rol de servicio podría interrumpir la funcionalidad de AWS Glue. Edita las funciones de servicio solo cuando AWS Glue te indique cómo hacerlo.

Para obtener instrucciones detalladas sobre cómo crear un rol de servicio para AWS Glue, consulte Paso 1: Crear una política de IAM para el servicio AWS Glue yPaso 2: creación de un rol de IAM para AWS Glue.

Funciones vinculadas al servicio para Glue AWS

Compatible con roles vinculados al servicio	No

Un rol vinculado a un servicio es un tipo de rol de servicio que está vinculado a un. Servicio de AWS El servicio puede asumir el rol para realizar una acción en su nombre. Los roles vinculados al servicio aparecen en usted Cuenta de AWS y son propiedad del servicio. Un administrador de IAM puede ver, pero no editar, los permisos de los roles vinculados a servicios.

Para más información sobre cómo crear o administrar roles vinculados a servicios, consulte Servicios de AWS que funcionan con IAM. Busque un servicio en la tabla que incluya `Yes` en la columna Rol vinculado a un servicio. Seleccione el vínculo Sí para ver la documentación acerca del rol vinculado a servicios para ese servicio.

Configuración de permisos de IAM para AWS Glue

AWS Identity and Access Management (IAM) se utiliza para definir políticas y roles que AWS Glue utiliza para acceder a recursos. Los siguientes pasos lo guiarán a través de varias opciones para configurar los permisos para AWS Glue. Según las necesidades de su negocio, es posible que tenga que añadir o reducir el acceso a los recursos.

> **Note**
>
> Para empezar a utilizar los permisos de IAM básicos para AWS Glue en su lugar, consulte Configuración de permisos de IAM para AWS Glue.

1. Crear una política de IAM para el servicio AWS Glue: cree una política de servicio que le permita acceder a los recursos de AWS Glue.
2. Crear un rol de IAM para AWS Glue: cree un rol de IAM y adjunte la política de servicio de AWS Glue y una política para los recursos de Amazon Simple Storage Service (Amazon S3) utilizados por AWS Glue.
3. Adjuntar una política a los usuarios o los grupos que acceden a AWS Glue: adjunte políticas a todo usuario o grupo que inicie sesión en la consola de AWS Glue.
4. Crear una política de IAM para los blocs de notas: cree una política de servidor de blocs de notas para utilizarla en la creación de servidores de blocs de notas en los puntos de conexión de desarrollo.
5. Crear un rol de IAM para los blocs de notas: cree un rol de IAM y adjunte la política de servidor de blocs de notas.
6. Crear una política de IAM para los blocs de notas de Amazon SageMaker: cree una política de IAM para utilizarla al crear blocs de notas de Amazon SageMaker en puntos de conexión de desarrollo.
7. Crear un rol de IAM para los blocs de notas de Amazon SageMaker: cree un rol de IAM y adjunte la política para otorgar permisos al crear blocs de notas de Amazon SageMaker en puntos de conexión de desarrollo.

Paso 1: Crear una política de IAM para el servicio AWS Glue

Para cualquier operación que obtenga acceso a datos que estén en otro recurso de AWS, como el acceso a sus objetos en Amazon S3, AWS Glue necesita permiso para obtener acceso al recurso en su nombre. Estos permisos los concede utilizando AWS Identity and Access Management (IAM).

> **Note**
>
> Puede omitir este paso si utiliza la política administrada de AWS, **`AWSGlueServiceRole`**.

En este paso, crea una política que es similar a `AWSGlueServiceRole`. Puede encontrar la versión más actualizada de `AWSGlueServiceRole` en la consola de IAM.

Para crear una política de IAM para AWS Glue

Esta política concede permiso para que algunas acciones de Amazon S3 administren recursos de su cuenta que AWS Glue necesita cuando asume el rol con esta política. Algunos de los recursos especificados en esta política hacen referencia a nombres predeterminados que AWS Glue utiliza para buckets de Amazon S3, scripts de ETL de Amazon S3, CloudWatch Logs y recursos de Amazon EC2. De forma predeterminada y para mayor simplicidad, AWS Glue escribe algunos objetos de Amazon S3 en buckets de su cuenta con el prefijo `aws-glue-*`.

1. Inicie sesión en la AWS Management Console y abra la consola de IAM en https://console.aws.amazon.com/iam/.
2. En el panel de navegación izquierdo, elija Policies (Políticas).
3. Elija Crear política.
4. En la pantalla Create Policy (Crear política) vaya a una pestaña para editar JSON. Cree un documento de política con las instrucciones JSON siguientes y, a continuación, elija Review policy (Revisar política).

   > **Note**
   >
   > Agregue todos los permisos necesarios para los recursos de Amazon S3. Es posible que le interese que la sección de recursos de su política de acceso abarque solo los recursos necesarios.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:*",
                "s3:GetBucketLocation",
                "s3:ListBucket",
                "s3:ListAllMyBuckets",
                "s3:GetBucketAcl",
                "ec2:DescribeVpcEndpoints",
                "ec2:DescribeRouteTables",
                "ec2:CreateNetworkInterface",
                "ec2:DeleteNetworkInterface",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcAttribute",
                "iam:ListRolePolicies",
                "iam:GetRole",
                "iam:GetRolePolicy",
                "cloudwatch:PutMetricData"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:CreateBucket",
                "s3:PutBucketPublicAccessBlock"
            ],
            "Resource": [
                "arn:aws:s3:::aws-glue-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
```

```
                "s3:PutObject",
                "s3:DeleteObject"
            ],
            "Resource": [
                "arn:aws:s3:::aws-glue-*/*",
                "arn:aws:s3:::*/*aws-glue-*/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::crawler-public*",
                "arn:aws:s3:::aws-glue-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:AssociateKmsKey"
            ],
            "Resource": [
                "arn:aws:logs:*:*:log-group:/aws-glue/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags",
                "ec2:DeleteTags"
            ],
            "Condition": {
                "ForAllValues:StringEquals": {
                    "aws:TagKeys": [
                        "aws-glue-service-resource"
                    ]
                }
            },
            "Resource": [
```

```
                "arn:aws:ec2:*:*:network-interface/*",
                "arn:aws:ec2:*:*:security-group/*",
                "arn:aws:ec2:*:*:instance/*"
            ]
        }
    ]
}
```

En la siguiente tabla se describen los permisos que esta política concede.

Action	Resource	Descripción
`"glue:*"`	`"*"`	Concede permiso para ejecutar todas las operaciones de la API de AWS Glue.
`"s3:GetBucketLocation", "s3:ListBucket", "s3:ListAllMyBuckets", "s3:GetBucketAcl",`	`"*"`	Permite crear listados de buckets de Amazon S3 a partir de rastreadores, trabajos, puntos de enlace de desarrollo y servidores de blocs de notas.
`"ec2:DescribeVpcEndpoints", "ec2:DescribeRouteTables", "ec2:CreateNetworkInterface", "ec2:DeleteNetworkInterface", "ec2:DescribeNetworkInterfaces", "ec2:DescribeSecurityGroups", "ec2:DescribeSubnets", "ec2:DescribeVpcAttribute",`	`"*"`	Permite configurar elementos de red de Amazon EC2, como nubes privadas virtuales (VPC), cuando se ejecutan trabajos, rastreadores y puntos de enlace de desarrollo.

Action	Resource	Descripción
`"iam:ListRolePolicies", "iam:GetRole", "iam:GetRolePolicy"`	`"*"`	Permite crear listados de roles de IAM a partir de rastreadores, trabajos, puntos de enlace de desarrollo y servidores de blocs de notas.
`"cloudwatch:PutMetricData"`	`"*"`	Permite escribir métricas de CloudWatch para trabajos.
`"s3:CreateBucket", "s3:PutBucketPublicAccessBlock"`	`"arn:aws:s3:::aws-glue-*"`	Permite la creación de buckets de Amazon S3 en su cuenta a partir de trabajos y servidores de blocs de notas. Convención de denominación: utiliza carpetas de Amazon S3 denominadas aws-glue-. Habilita AWS Glue para crear buckets que bloqueen el acceso público.

Action	Resource	Descripción
`"s3:GetObject", "s3:PutObject", "s3:DeleteObject"`	`"arn:aws:s3:::aws-glue-*/*", "arn:aws:s3:::*/*aws-glue-*/*"`	Permite obtener, colocar y eliminar objetos de Amazon S3 en su cuenta al almacenar objetos como scripts de ETL y ubicaciones de servidores de blocs de notas. Convención de denominación: concede permiso a buckets o carpetas de Amazon S3 cuyos nombres tienen el prefijo aws-glue-.
`"s3:GetObject"`	`"arn:aws:s3:::crawler-public*", "arn:aws:s3:::aws-glue-*"`	Permite conseguir objetos de Amazon S3 que utilizan los ejemplos y tutoriales de los rastreadores y los trabajos. Convención de denominación: los nombres de bucket de Amazon S3 comienzan con crawler-public y aws-glue-.
`"logs:CreateLogGroup", "logs:CreateLogStream", "logs:PutLogEvents"`	`"arn:aws:logs:*:*:log-group:/aws-glue/*"`	Permite escribir registros en CloudWatch Logs. Convención de denominación: AWS Glue escribe registros en grupos de registros cuyos nombres comienzan por aws-glue.

Action	Resource	Descripción
`"ec2:CreateTags", "ec2:DeleteTags"`	`"arn:aws:ec2:*:*:network-interface/*", "arn:aws:ec2:*:*:security-group/*", "arn:aws:ec2:*:*:instance/*"`	Permite el etiquetado de recursos de Amazon EC2; creados para puntos de enlace de desarrollo. Convención de denominación: AWS Glue etiqueta interfaces de red de Amazon EC2, grupos de seguridad e instancias con aws-glue-service-resource.

5. En la pantalla Review Policy (Revisar política), escriba su Policy Name (Nombre de política); por ejemplo, GlueServiceRolePolicy. Escriba una descripción opcional y, cuando quede satisfecho con la política, elija Create Policy (Crear política).

Paso 2: creación de un rol de IAM para AWS Glue

Tiene que conceder sus permisos de rol de IAM que AWS Glue pueda asumir cuando llame a otros servicios en su nombre. Esto incluye el acceso a Amazon S3 para todos los orígenes, los destinos, los scripts y los directorios temporales que utilice con AWS Glue. Los rastreadores, los trabajos y los puntos de enlace de desarrollo necesitan permiso.

Estos permisos los concede utilizando AWS Identity and Access Management (IAM). Agregue una política al rol de IAM que transfiera a AWS Glue.

Para crear un rol de IAM para AWS Glue

1. Inicie sesión en AWS Management Console y abra la consola IAM en https://console.aws.amazon.com/iam/.
2. En el panel de navegación izquierdo, seleccione Roles.
3. Seleccione Crear rol.
4. Para el tipo de rol, elija AWS Service (Servicio de AWS); busque y elija Glue, y luego, Next: Permissions (Siguiente: permisos).
5. En la página Attach permissions policy (Adjuntar política de permisos), elija las políticas que contienen los permisos necesarios; por ejemplo, la política administrada por AWS,

`AWSGlueServiceRole`, para permisos de AWS Glue generales y la política administrada de AWS, AmazonS3FullAccess, para obtener acceso a los recursos de Amazon S3. A continuación, seleccione Next: Review.

> **Note**
>
> Asegúrese de que una de las políticas de este rol conceda permisos a sus orígenes y destinos de Amazon S3. Es posible que le interese proporcionar su propia política para obtener acceso a determinados recursos de Amazon S3. Los orígenes de datos requieren permisos `s3:ListBucket` y `s3:GetObject`. Los destinos de datos requieren permisos `s3:ListBucket`, `s3:PutObject` y `s3:DeleteObject`. Para obtener más información acerca de la creación de una política de Amazon S3 para sus recursos, consulte Especificación de recursos en una política. Para ver un ejemplo de política de Amazon S3, consulte Cómo escribir políticas de IAM: cómo conceder acceso a un bucket de Amazon S3.
>
> Si tiene previsto obtener acceso a orígenes y destinos de Amazon S3 cifrados con SSE-KMS, asocie una política que permita a los rastreadores, trabajos y puntos de enlace de desarrollo de AWS Glue descifrar los datos. Para obtener más información, consulte Proteger los datos utilizando cifrado del lado del servidor con claves administradas por AWS KMS (SSE-KMS).
>
> A continuación, se muestra un ejemplo.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "kms:Decrypt"
         ],
         "Resource":[
            "arn:aws:kms:*:account-id-without-hyphens:key/key-id"
         ]
      }
   ]
}
```

6. En Role Name (Nombre del rol), escriba un nombre para el rol, por ejemplo, `AWSGlueServiceRoleDefault`. Cree el rol con el nombre prefijado con la cadena

`AWSGlueServiceRole` para permitir que se transfiera el rol desde usuarios de la consola al servicio. Las políticas proporcionadas por AWS Glue esperan que los roles de servicio de IAM comiencen con `AWSGlueServiceRole`. De lo contrario, deberá agregar una política para permitir a sus usuarios que el permiso `iam:PassRole` para roles de IAM coincida con la convención de denominación. Seleccione Crear rol.

> **Note**
>
> Cuando se crea un cuaderno con un rol, ese rol se pasa a las sesiones interactivas para que se pueda utilizar el mismo rol en ambos lugares. Como tal, el permiso `iam:PassRole` debe formar parte de la política del rol.
> Cree una nueva política para el rol mediante el siguiente ejemplo. Reemplace el número de cuenta por el suyo y por el nombre del rol.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::090000000210:role/<role_name>"
        }
    ]
}
```

Paso 3: Adjuntar una política a los usuarios o los grupos que accedan a AWS Glue

El administrador debe asignar permisos a cualquier usuario, grupo o rol mediante la consola de AWS Glue o la AWS Command Line Interface (AWS CLI). Estos permisos los concede con AWS Identity and Access Management (IAM), mediante políticas. En este paso, se describe la asignación de permisos a usuarios o grupos.

Cuando termine este paso, el usuario o el grupo tendrán las siguientes políticas adjuntas:

- Política administrada de AWS, `AWSGlueConsoleFullAccess`, o política personalizada GlueConsoleAccessPolicy

- **AWSGlueConsoleSageMakerNotebookFullAccess**
- **CloudWatchLogsReadOnlyAccess**
- **AWSCloudFormationReadOnlyAccess**
- **AmazonAthenaFullAccess**

Para adjuntar una política insertada e integrarla a un usuario o un grupo

Puede adjuntar una política administrada de AWS o una política insertada a un usuario o un grupo para obtener acceso a la consola de AWS Glue. Algunos de los recursos especificados en esta política hacen referencia a nombres predeterminados que AWS Glue utiliza para buckets de Amazon S3, scripts de ETL de Amazon S3, CloudWatch Logs, AWS CloudFormation y recursos de Amazon EC2. De forma predeterminada y para mayor simplicidad, AWS Glue escribe algunos objetos de Amazon S3 en buckets de su cuenta con el prefijo `aws-glue-*`.

> **Note**
>
> Puede omitir este paso si utiliza la política administrada de AWS **AWSGlueConsoleFullAccess**.

> **Important**
>
> AWS Glue necesita permiso para asumir un rol que se utiliza para realizar flujo de trabajo en su nombre. Para ello, debe agregar los permisos **iam:PassRole** a los usuarios o los grupos de AWS Glue. Esta política concede permiso a roles que comienzan por `AWSGlueServiceRole` para roles de servicio de AWS Glue y `AWSGlueServiceNotebookRole` para roles que son necesarios cuando crea un servidor de blocs de notas. También puede crear su propia política para permisos `iam:PassRole` que sigue su convención de denominación.
> Según las mejores prácticas de seguridad, se recomienda restringir el acceso mediante políticas más estrictas para limitar aún más el acceso al bucket de Amazon S3 y grupos de registros de Amazon CloudWatch. Para ver un ejemplo de política de Amazon S3, consulte Cómo escribir políticas de IAM: cómo conceder acceso a un bucket de Amazon S3.

En este paso, crea una política que es similar a `AWSGlueConsoleFullAccess`. Puede encontrar la versión más actualizada de `AWSGlueConsoleFullAccess` en la consola de IAM.

1. Inicie sesión en la AWS Management Console y abra la consola de IAM en https://console.aws.amazon.com/iam/.
2. En el panel de navegación, elija Usuarios o Grupos de usuarios.
3. En la lista, seleccione el nombre del usuario o del grupo en el que integrará una política.
4. Elija la pestaña Permissions (Permisos) y, si es necesario, expanda la sección Permissions Policies (Políticas de permisos).
5. Elija el enlace Add Inline policy (Añadir política en línea).
6. En la pantalla Create Policy (Crear política) vaya a una pestaña para editar JSON. Cree un documento de política con las instrucciones JSON siguientes y, a continuación, elija Review policy (Revisar política).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:*",
                "redshift:DescribeClusters",
                "redshift:DescribeClusterSubnetGroups",
                "iam:ListRoles",
                "iam:ListUsers",
                "iam:ListGroups",
                "iam:ListRolePolicies",
                "iam:GetRole",
                "iam:GetRolePolicy",
                "iam:ListAttachedRolePolicies",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcs",
                "ec2:DescribeVpcEndpoints",
                "ec2:DescribeRouteTables",
                "ec2:DescribeVpcAttribute",
                "ec2:DescribeKeyPairs",
                "ec2:DescribeInstances",
                "rds:DescribeDBInstances",
                "rds:DescribeDBClusters",
                "rds:DescribeDBSubnetGroups",
                "s3:ListAllMyBuckets",
                "s3:ListBucket",
```

```
                "s3:GetBucketAcl",
                "s3:GetBucketLocation",
                "cloudformation:DescribeStacks",
                "cloudformation:GetTemplateSummary",
                "dynamodb:ListTables",
                "kms:ListAliases",
                "kms:DescribeKey",
                "cloudwatch:GetMetricData",
                "cloudwatch:ListDashboards"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::*/*aws-glue-*/*",
                "arn:aws:s3:::aws-glue-*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "tag:GetResources"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:CreateBucket",
                "s3:PutBucketPublicAccessBlock"           ],
            "Resource": [
                "arn:aws:s3:::aws-glue-*"
            ]
        },
        {
```

```
            "Effect": "Allow",
            "Action": [
                "logs:GetLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:*:*:/aws-glue/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudformation:CreateStack",
                "cloudformation:DeleteStack"
            ],
            "Resource": "arn:aws:cloudformation:*:*:stack/aws-glue*/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:RunInstances"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:instance/*",
                "arn:aws:ec2:*:*:key-pair/*",
                "arn:aws:ec2:*:*:image/*",
                "arn:aws:ec2:*:*:security-group/*",
                "arn:aws:ec2:*:*:network-interface/*",
                "arn:aws:ec2:*:*:subnet/*",
                "arn:aws:ec2:*:*:volume/*"
            ]
        },
        {
            "Action": [
                "iam:PassRole"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:iam::*:role/AWSGlueServiceRole*",
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": [
                        "glue.amazonaws.com"
                    ]
                }
            }
```

```
        },
        {
            "Action": [
                "iam:PassRole"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:iam::*:role/AWSGlueServiceNotebookRole*",
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": [
                        "ec2.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Action": [
                "iam:PassRole"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:iam::*:role/service-role/AWSGlueServiceRole*"
            ],
            "Condition": {
                "StringLike": {
                    "iam:PassedToService": [
                        "glue.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

En la siguiente tabla se describen los permisos que esta política concede.

Action	Resource	Descripción
`"glue:*"`	`"*"`	Concede permiso para ejecutar todas las operaciones de la API de AWS Glue. Si ya había creado la política sin la acción `"glue:*"`, debe añadir el siguiente permiso individual a la política: • "glue:ListCrawlers" • "glue:BatchGetCrawlers" • "glue:ListTriggers" • "glue:BatchGetTriggers" • "glue:ListDevEndpoints" • "glue:BatchGetDevEndpoints" • "glue:ListJobs" • "glue:BatchGetJobs"
`"redshift:DescribeClusters", "redshift:DescribeClusterSubnetGroups"`	`"*"`	Permite la creación de conexiones con Amazon Redshift.

Action	Resource	Descripción
`"iam:ListRoles", "iam:ListRolePolicies", "iam:GetRole", "iam:GetRolePolicy", "iam:ListAttachedRolePolicies"`	`"*"`	Permite crear listados de roles de IAM cuando se trabaja con rastreadores, trabajos, puntos de enlace de desarrollo y servidores de blocs de notas.
`"ec2:DescribeSecurityGroups", "ec2:DescribeSubnets", "ec2:DescribeVpcs", "ec2:DescribeVpcEndpoints", "ec2:DescribeRouteTables", "ec2:DescribeVpcAttribute", "ec2:DescribeKeyPairs", "ec2:DescribeInstances"`	`"*"`	Permite configurar elementos de red de Amazon EC2, como VPC, al ejecutar trabajos, rastreadores y puntos de enlace de desarrollo.
`"rds:DescribeDBInstances"`	`"*"`	Permite la creación de conexiones con Amazon RDS.
`"s3:ListAllMyBuckets", "s3:ListBucket", "s3:GetBucketAcl", "s3:GetBucketLocation"`	`"*"`	Permite crear listados de buckets de Amazon S3 cuando se trabaja con rastreadores, trabajos, puntos de enlace de desarrollo y servidores de blocs de notas.
`"dynamodb:ListTables"`	`"*"`	Permite crear listados de tablas de DynamoDB.

Action	Resource	Descripción
`"kms:ListAliases", "kms:DescribeKey"`	`"*"`	Permite trabajar con claves de KMS.
`"cloudwatch:GetMetricData", "cloudwatch:ListDashboards"`	`"*"`	Permite trabajar con métricas de CloudWatch.
`"s3:GetObject", "s3:PutObject"`	`"arn:aws:s3::: aws-glue-*/*", "arn:aws:s3::: */*aws-glue-*/*", "arn:aws:s3::: aws-glue-*"`	Permite obtener y colocar objetos de Amazon S3 en su cuenta al almacenar objetos como scripts de ETL y ubicaciones de servidores de blocs de notas. Convención de denominación: concede permiso a buckets o carpetas de Amazon S3 cuyos nombres tienen el prefijo aws-glue-.
`"tag:GetResources"`	`"*"`	Permite recuperar etiquetas de AWS.

Action	Resource	Descripción
`"s3:CreateBucket", "s3:PutBucketPublicAccessBlock"`	`"arn:aws:s3::: aws-glue-*"`	Permite crear un bucket de Amazon S3 en la cuenta para almacenar objetos, como scripts de ETL y ubicaciones de servidores de blocs de notas. Convención de denominación: concede permiso a buckets de Amazon S3 cuyos nombres tienen el prefijo aws-glue-. Habilita AWS Glue para crear buckets que bloqueen el acceso público.
`"logs:GetLogEvents"`	`"arn:aws:logs:*:*: / aws-glue/*"`	Permite la recuperación de CloudWatch Logs. Convención de denominación: AWS Glue escribe registros en grupos de registros cuyos nombres comienzan por aws-glue-.

Action	Resource	Descripción
`"cloudformation:CreateStack", "cloudformation:DeleteStack"`	`"arn:aws:cloudformation:*:*:stack/aws-glue*/*"`	Permite administrar pilas de AWS CloudFormation cuando trabaja con servidores de blocs de notas. Convención de denominación: AWS Glue crea pilas cuyos nombres comienzan por aws-glue.
`"ec2:RunInstances"`	`"arn:aws:ec2:*:*:instance/*", "arn:aws:ec2:*:*:key-pair/*", "arn:aws:ec2:*:*:image/*", "arn:aws:ec2:*:*:security-group/*", "arn:aws:ec2:*:*:network-interface/*", "arn:aws:ec2:*:*:subnet/*", "arn:aws:ec2:*:*:volume/*"`	Permite la ejecución de puntos de enlace de desarrollo y servidores de blocs de notas.
`"iam:PassRole"`	`"arn:aws:iam::*:role/ AWSGlueServiceRole*"`	Permite que AWS Glue asuma el permiso `PassRole` para roles que comienzan con `AWSGlueServiceRole` .

Action	Resource	Descripción
`"iam:PassRole"`	`"arn:aws:iam::*:role/ AWSGlueServiceNotebookRole*"`	Permite que Amazon EC2 asuma el permiso `PassRole` para roles que comienzan con `AWSGlueServiceNotebookRole` .
`"iam:PassRole"`	`"arn:aws:iam::*:role/service-role/AWSGlueServiceRole*"`	Permite que AWS Glue asuma el permiso `PassRole` para roles que comienzan con `service-role/AWSGlueServiceRole` .

7. En la pantalla Review Policy (Revisar política), escriba un nombre para la política; por ejemplo, GlueConsoleAccessPolicy. Cuando esté satisfecho con la política, seleccione Create policy (Crear política). Asegúrese de que no aparece ningún error en un cuadro rojo en la parte superior de la pantalla. Corrija todos los errores notificados.

> **Note**
>
> Si la opción Use autoformatting está seleccionada, la política se vuelve a formatear cada vez que abra una política o elija la opción Validate Policy.

Para asociar la política administrada AWSGlueConsoleFullAccess

Puede asociar la política `AWSGlueConsoleFullAccess` para proporcionar permisos que el usuario de la consola de AWS Glue necesita.

> **Note**
>
> Puede omitir este paso si ha creado su propia política para el acceso a la consola de AWS Glue.

1. Inicie sesión en la AWS Management Console y abra la consola de IAM en https://console.aws.amazon.com/iam/.
2. En el panel de navegación, seleccione Policies (Políticas).
3. En la lista de políticas, seleccione la casilla de verificación situada junto a AWSGlueConsoleFullAccess. Puede utilizar el menú Filter y el cuadro de búsqueda para filtrar la lista de políticas.
4. Seleccione Policy actions (Acciones de la política) y, a continuación, Attach (Adjuntar).
5. Seleccione el usuario al que asociará la política. Puede utilizar el menú Filter (Filtro) y el cuadro de búsqueda para filtrar la lista entidades principales. Después de seleccionar el usuario al que asociará la política, seleccione Attach policy (Asociar política).

Asocie la política administrada, **`AWSGlueConsoleSageMakerNotebookFullAccess`**

Puede asociar la política `AWSGlueConsoleSageMakerNotebookFullAccess` a un usuario para administrar blocs de notas de SageMaker creados en la consola de AWS Glue. Además de otros permisos de la consola de AWS Glue necesarios, esta política concede acceso a los recursos que se necesitan para administrar blocs de notas de SageMaker.

1. Inicie sesión en la AWS Management Console y abra la consola de IAM en https://console.aws.amazon.com/iam/.
2. En el panel de navegación, seleccione Policies (Políticas).
3. En la lista de políticas, seleccione la casilla de verificación situada junto a AWSGlueConsoleSageMakerNotebookFullAccess. Puede utilizar el menú Filter y el cuadro de búsqueda para filtrar la lista de políticas.
4. Seleccione Policy actions (Acciones de la política) y, a continuación, Attach (Adjuntar).
5. Seleccione el usuario al que asociará la política. Puede utilizar el menú Filter (Filtro) y el cuadro de búsqueda para filtrar la lista entidades principales. Después de seleccionar el usuario al que asociará la política, seleccione Attach policy (Asociar política).

Para asociar la política administrada CloudWatchLogsReadOnlyAccess

Puede asociar la política CloudWatchLogsReadOnlyAccess a un usuario para ver los registros que AWS Glue crea en la consola de CloudWatch Logs.

1. Inicie sesión en la AWS Management Console y abra la consola de IAM en https://console.aws.amazon.com/iam/.

2. En el panel de navegación, seleccione Policies (Políticas).
3. En la lista de políticas, seleccione la casilla de verificación situada junto a la política CloudWatchLogsReadOnlyAccess. Puede utilizar el menú Filter y el cuadro de búsqueda para filtrar la lista de políticas.
4. Seleccione Policy actions (Acciones de la política) y, a continuación, Attach (Adjuntar).
5. Seleccione el usuario al que asociará la política. Puede utilizar el menú Filter (Filtro) y el cuadro de búsqueda para filtrar la lista entidades principales. Después de seleccionar el usuario al que asociará la política, seleccione Attach policy (Asociar política).

Para asociar la política administrada AWSCloudFormationReadOnlyAccess

Puede asociar la política AWSCloudFormationReadOnlyAccess a un usuario para ver las pilas de AWS CloudFormation que AWS Glue utiliza en la consola de AWS CloudFormation.

1. Inicie sesión en la AWS Management Console y abra la consola de IAM en https://console.aws.amazon.com/iam/.
2. En el panel de navegación, seleccione Policies (Políticas).
3. En la lista de políticas, seleccione la casilla de verificación situada junto a AWSCloudFormationReadOnlyAccess. Puede utilizar el menú Filter y el cuadro de búsqueda para filtrar la lista de políticas.
4. Seleccione Policy actions (Acciones de la política) y, a continuación, Attach (Adjuntar).
5. Seleccione el usuario al que asociará la política. Puede utilizar el menú Filter (Filtro) y el cuadro de búsqueda para filtrar la lista entidades principales. Después de seleccionar el usuario al que asociará la política, seleccione Attach policy (Asociar política).

Para asociar la política administrada AmazonAthenaFullAccess

Puede asociar la política AmazonAthenaFullAccess a un usuario para ver datos de Amazon S3 en la consola de Athena.

1. Inicie sesión en la AWS Management Console y abra la consola de IAM en https://console.aws.amazon.com/iam/.
2. En el panel de navegación, seleccione Policies (Políticas).
3. En la lista de políticas, seleccione la casilla de verificación situada junto a AmazonAthenaFullAccess. Puede utilizar el menú Filter y el cuadro de búsqueda para filtrar la lista de políticas.

4. Seleccione Policy actions (Acciones de la política) y, a continuación, Attach (Adjuntar).
5. Seleccione el usuario al que asociará la política. Puede utilizar el menú Filter (Filtro) y el cuadro de búsqueda para filtrar la lista entidades principales. Después de seleccionar el usuario al que asociará la política, seleccione Attach policy (Asociar política).

Paso 4: Crear una política de IAM para servidores de blocs de notas

Si pretende utilizar blocs de notas con puntos de enlace de desarrollo, debe especificar permisos al crear el servidor de bloc de notas. Estos permisos los concede utilizando AWS Identity and Access Management (IAM).

Esta política concede permiso para que algunas acciones de Amazon S3 administren recursos de su cuenta que AWS Glue necesita cuando asume el rol con esta política. Algunos de los recursos que se especifican en esta política hacen referencia a nombres predeterminados que AWS Glue utiliza para buckets de Amazon S3, scripts de ETL de Amazon S3 y recursos de Amazon EC2. De forma predeterminada y para mayor simplicidad, AWS Glue escribe algunos objetos de Amazon S3 en buckets de su cuenta con el prefijo `aws-glue-*`.

> **Note**
>
> Puede omitir este paso si utiliza la política administrada de AWS, **`AWSGlueServiceNotebookRole`**.

En este paso, crea una política que es similar a `AWSGlueServiceNotebookRole`. Puede encontrar la versión más actualizada de `AWSGlueServiceNotebookRole` en la consola de IAM.

Para crear una política de IAM para blocs de notas

1. Inicie sesión en la AWS Management Console y abra la consola de IAM en https://console.aws.amazon.com/iam/.
2. En el panel de navegación izquierdo, elija Policies (Políticas).
3. Elija Crear política.
4. En la pantalla Create Policy (Crear política) vaya a una pestaña para editar JSON. Cree un documento de política con las instrucciones JSON siguientes y, a continuación, elija Review policy (Revisar política).

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "glue:CreateDatabase",
            "glue:CreatePartition",
            "glue:CreateTable",
            "glue:DeleteDatabase",
            "glue:DeletePartition",
            "glue:DeleteTable",
            "glue:GetDatabase",
            "glue:GetDatabases",
            "glue:GetPartition",
            "glue:GetPartitions",
            "glue:GetTable",
            "glue:GetTableVersions",
            "glue:GetTables",
            "glue:UpdateDatabase",
            "glue:UpdatePartition",
            "glue:UpdateTable",
            "glue:GetJobBookmark",
            "glue:ResetJobBookmark",
            "glue:CreateConnection",
            "glue:CreateJob",
            "glue:DeleteConnection",
            "glue:DeleteJob",
            "glue:GetConnection",
            "glue:GetConnections",
            "glue:GetDevEndpoint",
            "glue:GetDevEndpoints",
            "glue:GetJob",
            "glue:GetJobs",
            "glue:UpdateJob",
            "glue:BatchDeleteConnection",
            "glue:UpdateConnection",
            "glue:GetUserDefinedFunction",
            "glue:UpdateUserDefinedFunction",
            "glue:GetUserDefinedFunctions",
            "glue:DeleteUserDefinedFunction",
            "glue:CreateUserDefinedFunction",
            "glue:BatchGetPartition",
```

```
            "glue:BatchDeletePartition",
            "glue:BatchCreatePartition",
            "glue:BatchDeleteTable",
            "glue:UpdateDevEndpoint",
            "s3:GetBucketLocation",
            "s3:ListBucket",
            "s3:ListAllMyBuckets",
            "s3:GetBucketAcl"
         ],
         "Resource":[
            "*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetObject"
         ],
         "Resource":[
            "arn:aws:s3:::crawler-public*",
            "arn:aws:s3:::aws-glue*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:PutObject",
            "s3:DeleteObject"
         ],
         "Resource":[
            "arn:aws:s3:::aws-glue*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "ec2:CreateTags",
            "ec2:DeleteTags"
         ],
         "Condition":{
            "ForAllValues:StringEquals":{
               "aws:TagKeys":[
                  "aws-glue-service-resource"
               ]
```

```
            }
         },
         "Resource":[
            "arn:aws:ec2:*:*:network-interface/*",
            "arn:aws:ec2:*:*:security-group/*",
            "arn:aws:ec2:*:*:instance/*"
         ]
      }
   ]
}
```

En la siguiente tabla se describen los permisos que esta política concede.

Action	Resource	Descripción
`"glue:*"`	`"*"`	Concede permiso para ejecutar todas las operaciones de la API de AWS Glue.
`"s3:GetBucketLocation", "s3:ListBucket", "s3:ListAllMyBuckets", "s3:GetBucketAcl"`	`"*"`	Permite crear listados de los buckets de Amazon S3 a partir de servidores de blocs de notas.
`"s3:GetObject"`	`"arn:aws:s3:::crawler-public*", "arn:aws:s3:::aws-glue-*"`	Permite obtener objetos de Amazon S3 que utilizan los ejemplos y tutoriales de los blocs de notas. Convención de denominación: los nombres de bucket de Amazon S3 comienzan con crawler-public y aws-glue-.

Action	Resource	Descripción
`"s3:PutObject", "s3:DeleteObject"`	`"arn:aws:s3:::aws-glue*"`	Permite colocar y eliminar objetos de Amazon S3 en su cuenta desde blocs de notas. Convención de denominación: utiliza carpetas de Amazon S3 denominadas aws-glue.
`"ec2:CreateTags", "ec2:DeleteTags"`	`"arn:aws:ec2:*:*:network-interface/*", "arn:aws:ec2:*:*:security-group/*", "arn:aws:ec2:*:*:instance/*"`	Permite el etiquetado de recursos de Amazon EC2 creados para servidores de blocs de notas. Convención de denominación: AWS Glue etiqueta instancias Amazon EC2 con aws-glue-service-resource.

5. En la pantalla Review Policy (Revisar política), escriba su Policy Name (Nombre de política); por ejemplo, GlueServiceNotebookPolicyDefault. Escriba una descripción opcional y, cuando quede satisfecho con la política, elija Create Policy (Crear política).

Paso 5: Crear un rol de IAM para servidores de blocs de notas

Si pretende utilizar blocs de notas con puntos de enlace de desarrollo, debe conceder los permisos de rol de IAM. Estos permisos los concede al utilizar IAM de AWS Identity and Access Management, mediante un rol de IAM.

> **Note**
>
> Cuando se crea un rol de IAM utilizando la consola de IAM, esta crea automáticamente un perfil de instancia y le da el mismo nombre que el rol al que corresponde.

Para crear un rol de IAM para blocs de notas

1. Inicie sesión en AWS Management Console y abra la consola IAM en https://console.aws.amazon.com/iam/.
2. En el panel de navegación izquierdo, seleccione Roles.
3. Seleccione Crear rol.
4. Para el tipo de rol, elija AWS Service (Servicio de AWS), encuentre y elija EC2 y seleccione el caso de uso de EC2; por último, elija Next: Permissions (Siguiente: Permisos).
5. En la página Attach permissions policy (Adjuntar política de permisos), elija las políticas que contienen los permisos necesarios; por ejemplo, AWSGlueServiceNotebookRole, para permisos de AWS Glue generales, y la política administrada por AWS, AmazonS3FullAccess, para obtener acceso a recursos de Amazon S3. A continuación, seleccione Next: Review.

> **Note**
>
> Asegúrese de que una de las políticas de este rol conceda permisos a sus orígenes y destinos de Amazon S3. También confirme que su política le permite obtener acceso completo a la ubicación en la que almacena su bloc de notas cuando crea un servidor de blocs de notas. Es posible que le interese proporcionar su propia política para obtener acceso a determinados recursos de Amazon S3. Para obtener más información acerca de la creación de una política de Amazon S3 para sus recursos, consulte Especificación de recursos en una política.
> Si tiene previsto obtener acceso a orígenes y destinos de Amazon S3 cifrados con SSE-KMS, asocie una política que permita a los blocs de notas descifrar los datos. Para obtener más información, consulte Proteger los datos utilizando cifrado del lado del servidor con claves administradas por AWS KMS (SSE-KMS).
> A continuación, se muestra un ejemplo.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "kms:Decrypt"
         ],
         "Resource":[
            "arn:aws:kms:*:account-id-without-hyphens:key/key-id"
```

```
            ]
        }
    ]
}
```

6. Escriba un nombre para el rol en Nombre de rol. Cree el rol con el nombre prefijado con la cadena `AWSGlueServiceNotebookRole` para permitir que se transfiera el rol desde usuarios de la consola al servidor de blocs de notas. Las políticas proporcionadas por AWS Glue esperan que los roles de servicio de IAM comiencen con `AWSGlueServiceNotebookRole`. De lo contrario, deberá agregar una política a sus usuarios para permitir que el permiso `iam:PassRole` para roles de IAM coincida con su convención de denominación. Por ejemplo, escriba `AWSGlueServiceNotebookRoleDefault`. A continuación, elija Crear rol.

Paso 6: Crear una política de IAM para blocs de notas de SageMaker

Si pretende utilizar blocs de notas de SageMaker con puntos de enlace de desarrollo, debe especificar los permisos al crear el bloc de notas. Estos permisos los concede utilizando AWS Identity and Access Management (IAM).

Para crear una política de IAM para blocs de notas de SageMaker

1. Inicie sesión en la AWS Management Console y abra la consola de IAM en https://console.aws.amazon.com/iam/.
2. En el panel de navegación izquierdo, elija Policies (Políticas).
3. Elija Crear política.
4. En la página Create Policy (Crear política) vaya a una pestaña para editar JSON. Cree un documento de política con las siguientes instrucciones JSON. Modifique el *nombre-del-bucket*, el *código-de-región* y el *identificador-de-cuenta* del entorno.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:ListBucket"
            ],
            "Effect": "Allow",
            "Resource": [
```

```
                "arn:aws:s3:::bucket-name"
            ]
        },
        {
            "Action": [
                "s3:GetObject"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:s3:::bucket-name*"
            ]
        },
        {
            "Action": [
                "logs:CreateLogStream",
                "logs:DescribeLogStreams",
                "logs:PutLogEvents",
                "logs:CreateLogGroup"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:logs:region-code:account-id:log-group:/aws/sagemaker/*",
                "arn:aws:logs:region-code:account-id:log-group:/aws/sagemaker/
*:log-stream:aws-glue-*"
            ]
        },
        {
            "Action": [
                "glue:UpdateDevEndpoint",
                "glue:GetDevEndpoint",
                "glue:GetDevEndpoints"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:glue:region-code:account-id:devEndpoint/*"
            ]
        },
        {
            "Action": [
                "sagemaker:ListTags"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:sagemaker:region-code:account-id:notebook-instance/*"
```

```
            ]
        }
    ]
}
```

A continuación, elija Review policy (Revisar política).

En la siguiente tabla se describen los permisos que esta política concede.

Action	Resource	Descripción
`"s3:ListBucket*"`	`"arn:aws:s3::: bucket-name "`	Concede permisos para enumerar los buckets de Amazon S3.
`"s3:GetObject"`	`"arn:aws:s3::: bucket-name *"`	Concede permiso para obtener objetos de Amazon S3 que se utilizan en blocs de notas de SageMaker.
`"logs:CreateLogStream", "logs:DescribeLogStreams", "logs:PutLogEvents", "logs:CreateLogGroup"`	`"arn:aws:logs: region-code :account-id :log-group:/aws/sagemaker/*", "arn:aws:logs: region-code :account-id :log-group:/aws/sagemaker/*:log-stream:aws-glue-*"`	Concede permiso para escribir registros en Amazon CloudWatch Logs desde blocs de notas. Convención de denominación: escribe en grupos de registros cuyos nombres comienzan por aws-glue.
`"glue:UpdateDevEndpoint", "glue:GetDevEndpoint", "glue:GetDevEndpoints"`	`"arn:aws:glue: region-code :account-id :devEndpoint/*"`	Concede permiso para utilizar un punto de enlace de desarrollo de blocs de notas de SageMaker.

Action	Resource	Descripción
`"sagemaker:ListTags"`	`"arn:aws:sagemaker:region-code:account-id:notebook-instance/*"`	Concede permiso para devolver etiquetas de un recurso de SageMaker. La etiqueta `aws-glue-dev-endpoint` es necesaria en el bloc de notas de SageMaker para conectar este bloc de notas a un punto de enlace de desarrollo.

5. En la pantalla Review Policy (Revisar política), escriba su Policy Name (Nombre de política); por ejemplo, `AWSGlueSageMakerNotebook`. Escriba una descripción opcional y, cuando quede satisfecho con la política, elija Create Policy (Crear política).

Paso 7: Crear un rol de IAM para blocs de notas de SageMaker

Si pretende utilizar blocs de notas de SageMaker con puntos de enlace de desarrollo, debe conceder los permisos de rol de IAM. Estos permisos los concede al utilizar AWS Identity and Access Management (IAM), mediante un rol de IAM.

Para crear un rol de IAM para blocs de notas de SageMaker

1. Inicie sesión en AWS Management Console y abra la consola IAM en https://console.aws.amazon.com/iam/.
2. En el panel de navegación izquierdo, seleccione Roles.
3. Seleccione Crear rol.
4. Para el tipo de rol, elija AWS Service (Servicio de AWS), busque y elija SageMaker y seleccione el caso de uso SageMaker - Execution (SageMaker: ejecución). A continuación, elija Siguiente: permisos.
5. En la página Attach permissions policy (Asociar la política de permisos), elija las políticas que contengan los permisos necesarios; por ejemplo, AmazonSageMakerFullAccess. Elija Siguiente: Revisar.

   Si tiene previsto obtener acceso a orígenes y destinos de Amazon S3 que se cifran con SSE-KMS, asocie una política que permita a los blocs de notas descifrar los datos, como se muestra

en el siguiente ejemplo. Para obtener más información, consulte Proteger los datos utilizando cifrado del lado del servidor con claves administradas por AWS KMS (SSE-KMS).

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "kms:Decrypt"
         ],
         "Resource":[
            "arn:aws:kms:*:account-id-without-hyphens:key/key-id"
         ]
      }
   ]
}
```

6. Escriba un nombre para el rol en Nombre de rol. Para permitir que se transfiera el rol desde usuarios de la consola a SageMaker, use un nombre que tenga como prefijo la cadena `AWSGlueServiceSageMakerNotebookRole`. Las políticas proporcionadas por AWS Glue esperan que los roles de IAM comiencen con `AWSGlueServiceSageMakerNotebookRole`. De lo contrario, deberá agregar una política a sus usuarios para permitir que el permiso `iam:PassRole` para roles de IAM coincida con su convención de denominación.

   Por ejemplo, ingrese `AWSGlueServiceSageMakerNotebookRole-Default`, y luego elija Create role (Crear rol).

7. Después de crear el rol, asocie la política que permite los permisos adicionales necesarios para crear blocs de notas de SageMaker desde AWS Glue.

   Abra el rol que acaba de crear, `AWSGlueServiceSageMakerNotebookRole-Default`, y elija Attach policies (Asociar políticas). Asocie la política que creó, denominada `AWSGlueSageMakerNotebook`, al rol.

Ejemplos de políticas de control de acceso de AWS Glue

En esta sección se incluyen ejemplos de las políticas de control de acceso basadas en identidad (IAM) y las políticas de recursos de AWS Glue.

Contenido

- Ejemplos de políticas basadas en identidades para AWS Glue
  - Prácticas recomendadas sobre las políticas
  - Los permisos de nivel de recursos solo se aplican a objetos específicos de AWS Glue.
  - Uso de la consola de AWS Glue
  - Cómo permitir a los usuarios consultar sus propios permisos
  - Conceder permiso de solo lectura a una tabla
  - Filtrar tablas por permiso GetTables
  - Conceder acceso completo a una tabla y todas las particiones
  - Controlar el acceso mediante prefijo de nombre y denegación explícita
  - Conceder acceso mediante etiquetas
  - Denegar acceso mediante etiquetas
  - Usar etiquetas con las operaciones de API List y Batch
  - Controlar la configuración mediante claves de condición o claves de contexto
    - Políticas que controlan la configuración mediante claves de condición
    - Políticas que controlan la configuración mediante claves de contexto
  - Cómo denegar a una identidad la capacidad de crear sesiones de vista previa de datos
- Ejemplos de políticas basadas en recursos para AWS Glue
  - Consideraciones para el uso de políticas basadas en recursos con AWS Glue
  - Uso de una política de recursos para controlar el acceso en la misma cuenta

Ejemplos de políticas basadas en identidades para AWS Glue

De forma predeterminada, los usuarios y los roles no tienen permiso para crear ni para modificar recursos de AWS Glue. Tampoco pueden realizar tareas mediante la AWS Management Console, la AWS Command Line Interface (AWS CLI) o la API de AWS. Un administrador de IAM puede crear políticas de IAM para conceder permisos a los usuarios para realizar acciones en los recursos que necesitan. A continuación, el administrador puede agregar las políticas de IAM a los roles, y los usuarios pueden asumirlos.

Para obtener información acerca de cómo crear una política basada en identidades de IAM mediante el uso de estos documentos de políticas JSON de ejemplo, consulte Creación de políticas de IAM en la Guía del usuario de IAM.

Para obtener más información sobre las acciones y los tipos de recursos definidos por AWS Glue, incluido el formato de los ARN para cada tipo de recurso, consulte Acciones, recursos y claves de condición para AWS Glue en la Referencia de autorizaciones de servicio.

> **Note**
>
> Todos los ejemplos que se proporcionan en esta sección utilizan la región `us-west-2`. Puede reemplazarla por la región de AWS que desee usar.

Temas

- Prácticas recomendadas sobre las políticas
- Los permisos de nivel de recursos solo se aplican a objetos específicos de AWS Glue.
- Uso de la consola de AWS Glue
- Cómo permitir a los usuarios consultar sus propios permisos
- Conceder permiso de solo lectura a una tabla
- Filtrar tablas por permiso GetTables
- Conceder acceso completo a una tabla y todas las particiones
- Controlar el acceso mediante prefijo de nombre y denegación explícita
- Conceder acceso mediante etiquetas
- Denegar acceso mediante etiquetas
- Usar etiquetas con las operaciones de API List y Batch
- Controlar la configuración mediante claves de condición o claves de contexto
- Cómo denegar a una identidad la capacidad de crear sesiones de vista previa de datos

Prácticas recomendadas sobre las políticas

Las políticas basadas en identidades determinan si alguien puede crear o eliminar los recursos de AWS Glue en la cuenta y también si pueden acceder a ellos. Estas acciones pueden generar costes adicionales para su Cuenta de AWS. Siga estas directrices y recomendaciones al crear o editar políticas basadas en identidades:

- Comience con las políticas administradas por AWSy continúe con los permisos de privilegio mínimo: a fin de comenzar a conceder permisos a los usuarios y las cargas de trabajo, utilice las políticas administradas por AWS, que conceden permisos para muchos casos de uso comunes.

Están disponibles en su Cuenta de AWS. Se recomienda definir políticas administradas por el cliente de AWS específicas para sus casos de uso a fin de reducir aún más los permisos. Con el fin de obtener más información, consulte las políticas administradas por AWS o las políticas administradas por AWS para funciones de trabajo en la Guía de usuario de IAM.

- Aplique permisos de privilegio mínimo: cuando establezca permisos con políticas de IAM, conceda solo los permisos necesarios para realizar una tarea. Para ello, debe definir las acciones que se pueden llevar a cabo en determinados recursos en condiciones específicas, también conocidos como permisos de privilegios mínimos. Con el fin de obtener más información sobre el uso de IAM para aplicar permisos, consulte Políticas y permisos en IAM en la Guía del usuario de IAM.
- Utilice condiciones en las políticas de IAM para restringir aún más el acceso: puede agregar una condición a sus políticas para limitar el acceso a las acciones y los recursos. Por ejemplo, puede escribir una condición de políticas para especificar que todas las solicitudes deben enviarse utilizando SSL. También puede usar condiciones para conceder acceso a acciones de servicios si se emplean a través de un Servicio de AWS determinado, como por ejemplo AWS CloudFormation. Para obtener más información, consulte Elementos de la política de JSON de IAM: Condición en la Guía del usuario de IAM.
- Utilice el analizador de acceso de IAM para validar las políticas de IAM con el fin de garantizar la seguridad y funcionalidad de los permisos: el analizador de acceso de IAM valida políticas nuevas y existentes para que respeten el lenguaje (JSON) de las políticas de IAM y las prácticas recomendadas de IAM. El analizador de acceso de IAM proporciona más de 100 verificaciones de políticas y recomendaciones procesables para ayudar a crear políticas seguras y funcionales. Para más información, consulte Política de validación de Analizador de acceso de IAM en la Guía de usuario de IAM.
- Solicite la autenticación multifactor (MFA): si se encuentra en una situación en la que necesita usuarios raíz o de IAM en su Cuenta de AWS, active la MFA para mayor seguridad. Para solicitar la MFA cuando se invocan las operaciones de la API, agregue las condiciones de la MFA a sus políticas. Para más información, consulte Configuración del acceso a una API protegido por MFA en la Guía de usuario de IAM.

Para obtener más información sobre las prácticas recomendadas de IAM, consulte las Prácticas recomendadas de seguridad en IAM en la Guía del usuario de IAM.

Los permisos de nivel de recursos solo se aplican a objetos específicos de AWS Glue.

Solo puede definir un control detallado para los objetos específicos de AWS Glue. Por lo tanto, debe escribir la política de IAM de sus clientes de modo que las operaciones API que permiten utilizar

nombres de recursos de Amazon (Amazon Resource Names, ARN) con la instrucción `Resource` no se mezclen con las operaciones API que no permiten el uso de ARN.

Por ejemplo, la siguiente política de IAM permite operaciones de la API para `GetClassifier` y `GetJobRun`. Define el `Resource` como * porque AWS Glue no permite ARN para clasificadores y ejecuciones de trabajo. Dado que se permite utilizar ARN con determinadas operaciones API, como `GetDatabase` y `GetTable`, se pueden especificar ARN en la segunda mitad de la política.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:GetClassifier*",
                "glue:GetJobRun*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "glue:Get*"
            ],
            "Resource": [
                "arn:aws:glue:us-east-1:123456789012:catalog",
                "arn:aws:glue:us-east-1:123456789012:database/default",
                "arn:aws:glue:us-east-1:123456789012:table/default/e*1*",
                "arn:aws:glue:us-east-1:123456789012:connection/connection2"
            ]
        }
    ]
}
```

Para ver una lista de objetos de AWS Glue que permiten el uso de ARN, consulte ARN del recurso.

Uso de la consola de AWS Glue

Para acceder a la consola de AWS Glue, debe tener un conjunto mínimo de permisos. Estos permisos deben permitirle enumerar y consultar detalles sobre los recursos de AWS Glue en su Cuenta de AWS. Si crea una política basada en identidades que sea más restrictiva que el mínimo

de permisos necesarios, la consola no funcionará del modo esperado para las entidades (usuarios o roles) que tengan esa política.

No es necesario que conceda permisos mínimos para la consola a los usuarios que solo realizan llamadas a la AWS CLI o a la API de AWS. En su lugar, permite acceso únicamente a las acciones que coincidan con la operación de API que intentan realizar.

Para asegurarse de que los usuarios y los roles puedan seguir utilizando la consola de AWS Glue, también asocie a las entidades la política administrada por AWS `ConsoleAccess` o `ReadOnly` de AWS Glue. Para más información, consulte Adición de permisos a un usuario en la Guía del usuario de IAM:

Para que un usuario pueda trabajar con la consola de AWS Glue, debe tener un conjunto mínimo de permisos que le permitan trabajar con los recursos de AWS Glue en su cuenta de AWS. Además de estos permisos de AWS Glue, la consola requiere permisos de los servicios a continuación:

- Permisos de Amazon CloudWatch Logs para mostrar registros.
- Permisos de AWS Identity and Access Management (IAM) para enumerar y transferir roles.
- Permisos AWS CloudFormation para trabajar con pilas.
- Permisos de Amazon Elastic Compute Cloud (Amazon EC2) para enumerar VPC, subredes, grupos de seguridad, instancias y otros objetos.
- Permisos de Amazon Simple Storage Service (Amazon S3) para enumerar buckets y objetos, y recuperar y guardar scripts.
- Permisos necesarios de Amazon Redshift para trabajar con clústeres.
- Permisos de Amazon Relational Database Service (Amazon RDS) para enumerar instancias.

Para obtener más información sobre los permisos que necesitan los usuarios para ver y trabajar con la consola de AWS Glue, consulte Paso 3: Adjuntar una política a los usuarios o los grupos que accedan a AWS Glue.

Si crea una política de IAM que sea más restrictiva que el mínimo de permisos necesarios, la consola no funcionará del modo esperado para los usuarios con esa política de IAM. Para asegurarse de que esos usuarios puedan seguir usando la consola de AWS Glue, asocie también la política administrada `AWSGlueConsoleFullAccess`, según se explica en AWS políticas gestionadas (predefinidas) para AWS Glue.

Cómo permitir a los usuarios consultar sus propios permisos

En este ejemplo, se muestra cómo podría crear una política que permita a los usuarios de IAM ver las políticas administradas e insertadas que se asocian a la identidad de sus usuarios. Esta política incluye permisos para llevar a cabo esta acción en la consola o mediante programación con la AWS CLI o la API de AWS.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Conceder permiso de solo lectura a una tabla

La siguiente política concede permiso de solo lectura a una tabla `books` en la base de datos `db1`. Para obtener más información sobre nombres de recursos de Amazon (Amazon Resource Names, ARN), consulte ARN de Data Catalog.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "GetTablesActionOnBooks",
      "Effect": "Allow",
      "Action": [
        "glue:GetTables",
        "glue:GetTable"
      ],
      "Resource": [
        "arn:aws:glue:us-west-2:123456789012:catalog",
        "arn:aws:glue:us-west-2:123456789012:database/db1",
        "arn:aws:glue:us-west-2:123456789012:table/db1/books"
      ]
    }
  ]
}
```

Esta política concede permisos de solo lectura a una tabla denominada `books` en la base de datos denominada `db1`. Para conceder un permiso `Get` a una tabla, también se requiere permiso para el catálogo y los recursos de la base de datos.

La siguiente política concede los permisos mínimos necesarios crear una tabla `tbl` en la base de datos `db1`:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "glue:CreateTable"
      ],
      "Resource": [
        "arn:aws:glue:us-west-2:123456789012:table/db1/tbl1",
```

```
        "arn:aws:glue:us-west-2:123456789012:database/db1",
        "arn:aws:glue:us-west-2:123456789012:catalog"
      ]
    }
  ]
}
```

Filtrar tablas por permiso GetTables

Supongamos que hay tres tablas: `customers`, `stores` y `store_sales` en base de datos db1. La siguiente política concede permiso `GetTables` a `stores` y `store_sales`, pero no a `customers`. Cuando llama a `GetTables` con esta política, el resultado contiene únicamente las dos tablas autorizadas (la tabla `customers` no se devuelve).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "GetTablesExample",
            "Effect": "Allow",
            "Action": [
                "glue:GetTables"
            ],
            "Resource": [
                "arn:aws:glue:us-west-2:123456789012:catalog",
                "arn:aws:glue:us-west-2:123456789012:database/db1",
                "arn:aws:glue:us-west-2:123456789012:table/db1/store_sales",
                "arn:aws:glue:us-west-2:123456789012:table/db1/stores"
            ]
        }
    ]
}
```

Puede simplificar la política anterior mediante `store*` para hacer coincidir los nombres de las tablas que comienzan por `store`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "GetTablesExample2",
            "Effect": "Allow",
```

```
            "Action": [
                "glue:GetTables"
            ],
            "Resource": [
                "arn:aws:glue:us-west-2:123456789012:catalog",
                "arn:aws:glue:us-west-2:123456789012:database/db1",
                "arn:aws:glue:us-west-2:123456789012:table/db1/store*"
            ]
        }
    ]
}
```

Del mismo modo, utilizando `/db1/*` para que coincida con todas las tablas de `db1`, la siguiente política concede a `GetTables` acceso a todas las tablas de `db1`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "GetTablesReturnAll",
            "Effect": "Allow",
            "Action": [
                "glue:GetTables"
            ],
            "Resource": [
                "arn:aws:glue:us-west-2:123456789012:catalog",
                "arn:aws:glue:us-west-2:123456789012:database/db1",
                "arn:aws:glue:us-west-2:123456789012:table/db1/*"
            ]
        }
    ]
}
```

Si no se proporciona un ARN de tabla, se lleva a cabo correctamente una llamada a `GetTables`, pero devuelve una lista vacía.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "GetTablesEmptyResults",
            "Effect": "Allow",
```

```
            "Action": [
                "glue:GetTables"
            ],
            "Resource": [
                "arn:aws:glue:us-west-2:123456789012:catalog",
                "arn:aws:glue:us-west-2:123456789012:database/db1"
            ]
        }
    ]
}
```

Si el ARN de base de datos no se encuentra en la política, se produce un error con una llamada a `GetTables` con `AccessDeniedException`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "GetTablesAccessDeny",
            "Effect": "Allow",
            "Action": [
                "glue:GetTables"
            ],
            "Resource": [
                "arn:aws:glue:us-west-2:123456789012:catalog",
                "arn:aws:glue:us-west-2:123456789012:table/db1/*"
            ]
        }
    ]
}
```

Conceder acceso completo a una tabla y todas las particiones

La siguiente política concede todos los permisos a una tabla denominada `books` en la base de datos db1. Esto incluye permisos de lectura y escritura en la propia tabla, en las versiones archivadas de ella y en todas sus particiones.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "FullAccessOnTable",
```

```
            "Effect": "Allow",
            "Action": [
                "glue:CreateTable",
                "glue:GetTable",
                "glue:GetTables",
                "glue:UpdateTable",
                "glue:DeleteTable",
                "glue:BatchDeleteTable",
                "glue:GetTableVersion",
                "glue:GetTableVersions",
                "glue:DeleteTableVersion",
                "glue:BatchDeleteTableVersion",
                "glue:CreatePartition",
                "glue:BatchCreatePartition",
                "glue:GetPartition",
                "glue:GetPartitions",
                "glue:BatchGetPartition",
                "glue:UpdatePartition",
                "glue:DeletePartition",
                "glue:BatchDeletePartition"
            ],
            "Resource": [
                        "arn:aws:glue:us-west-2:123456789012:catalog",
                        "arn:aws:glue:us-west-2:123456789012:database/db1",
                        "arn:aws:glue:us-west-2:123456789012:table/db1/books"
                        ]
        }
    ]
}
```

La política anterior se puede simplificar en la práctica.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "FullAccessOnTable",
            "Effect": "Allow",
            "Action": [
                "glue:*Table*",
                "glue:*Partition*"
            ],
            "Resource": [
```

```
                "arn:aws:glue:us-west-2:123456789012:catalog",
                "arn:aws:glue:us-west-2:123456789012:database/db1",
                "arn:aws:glue:us-west-2:123456789012:table/db1/books"
            ]
        }
    ]
}
```

Tenga en cuenta que la granularidad mínima del control de acceso detallado se encuentra en el nivel de tabla. Esto significa que no puede conceder a un usuario acceso a algunas particiones de una tabla pero no a otras, o a algunas columnas de la tabla pero no a otras. Un usuario tiene acceso a toda una tabla o a nada de ella.

Controlar el acceso mediante prefijo de nombre y denegación explícita

En este ejemplo, supongamos que las bases de datos y las tablas de su AWS Glue Data Catalog están organizadas con prefijos de nombre. Las bases de datos en la fase de desarrollo tienen el prefijo de nombre `dev-` y las que están en fase de producción tienen el prefijo de nombre `prod-`. Utilice la siguiente política para conceder a los desarrolladores acceso completo a todas las bases de datos, tablas, UDF, etc., que tengan el prefijo `dev-`. Sin embargo, debe conceder acceso de solo lectura a todo lo que incluya el prefijo `prod-`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DevAndProdFullAccess",
            "Effect": "Allow",
            "Action": [
                "glue:*Database*",
                "glue:*Table*",
                "glue:*Partition*",
                "glue:*UserDefinedFunction*",
                "glue:*Connection*"
            ],
            "Resource": [
                "arn:aws:glue:us-west-2:123456789012:catalog",
                "arn:aws:glue:us-west-2:123456789012:database/dev-*",
                "arn:aws:glue:us-west-2:123456789012:database/prod-*",
                "arn:aws:glue:us-west-2:123456789012:table/dev-*/*",
                "arn:aws:glue:us-west-2:123456789012:table/*/dev-*",
                "arn:aws:glue:us-west-2:123456789012:table/prod-*/*",
```

```
                "arn:aws:glue:us-west-2:123456789012:table/*/prod-*",
                "arn:aws:glue:us-west-2:123456789012:userDefinedFunction/dev-*/*",
                "arn:aws:glue:us-west-2:123456789012:userDefinedFunction/*/dev-*",
                "arn:aws:glue:us-west-2:123456789012:userDefinedFunction/prod-*/*",
                "arn:aws:glue:us-west-2:123456789012:userDefinedFunction/*/prod-*",
                "arn:aws:glue:us-west-2:123456789012:connection/dev-*",
                "arn:aws:glue:us-west-2:123456789012:connection/prod-*"
            ]
        },
        {
            "Sid": "ProdWriteDeny",
            "Effect": "Deny",
            "Action": [
                "glue:*Create*",
                "glue:*Update*",
                "glue:*Delete*"
            ],
            "Resource": [
                "arn:aws:glue:us-west-2:123456789012:database/prod-*",
                "arn:aws:glue:us-west-2:123456789012:table/prod-*/*",
                "arn:aws:glue:us-west-2:123456789012:table/*/prod-*",
                "arn:aws:glue:us-west-2:123456789012:userDefinedFunction/prod-*/*",
                "arn:aws:glue:us-west-2:123456789012:userDefinedFunction/*/prod-*",
                "arn:aws:glue:us-west-2:123456789012:connection/prod-*"
            ]
        }
    ]
}
```

La segunda instrucción de la política anterior utiliza `deny` explícito. Puede utilizar `deny` explícito para sobrescribir los permisos `allow` que se conceden al principal. Esto le permite bloquear el acceso a los recursos de vital importancia y evitar que otra política conceda accidentalmente acceso a ellos.

En el ejemplo anterior, aunque la primera instrucción concede acceso completo a los recursos de `prod-`, la segunda instrucción revoca explícitamente el acceso de escritura a ellos, lo que deja acceso de solo lectura a los recursos de `prod-`.

Conceder acceso mediante etiquetas

Por ejemplo, supongamos que desea limitar el acceso de un usuario específico de la cuenta denominado `Tom` a un desencadenador `t2`. Todos los demás usuarios, entre los que se incluye

Sam, tienen acceso para desencadenar t1. Los desencadenadores t1 y t2 tienen las siguientes propiedades.

```
aws glue get-triggers
{
    "Triggers": [
        {
            "State": "CREATED",
            "Type": "SCHEDULED",
            "Name": "t1",
            "Actions": [
                {
                    "JobName": "j1"
                }
            ],
            "Schedule": "cron(0 0/1 * * ? *)"
        },
        {
            "State": "CREATED",
            "Type": "SCHEDULED",
            "Name": "t2",
            "Actions": [
                {
                    "JobName": "j1"
                }
            ],
            "Schedule": "cron(0 0/1 * * ? *)"
        }
    ]
}
```

El administrador de AWS Glue asoció un valor de etiqueta Tom (aws:ResourceTag/Name": "Tom") al desencadenador t2. El administrador de AWS Glue también le proporcionó a Tom una política de IAM con una declaración de condición basada en la etiqueta. Por tanto, Tom solamente puede utilizar una operación de AWS Glue que actúe sobre los recursos con el valor de etiqueta Tom.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
         "Effect": "Allow",
         "Action": "glue:*",
```

```
         "Resource": "*",
         "Condition": {
             "StringEquals": {
                "aws:ResourceTag/Name": "Tom"
              }
          }
         }
    ]
}
```

Cuando Tom intenta obtener acceso al desencadenador `t1`, recibe un mensaje de acceso denegado. Sin embargo, puede recuperar correctamente el desencadenador `t2`.

```
aws glue get-trigger --name t1

An error occurred (AccessDeniedException) when calling the GetTrigger operation:
 User: Tom is not authorized to perform: glue:GetTrigger on resource: arn:aws:glue:us-
east-1:123456789012:trigger/t1

aws glue get-trigger --name t2
{
    "Trigger": {
        "State": "CREATED",
        "Type": "SCHEDULED",
        "Name": "t2",
        "Actions": [
            {
                "JobName": "j1"
            }
        ],
        "Schedule": "cron(0 0/1 * * ? *)"
    }
}
```

Tom no puede usar la operación de API `GetTriggers` plural para mostrar los desencadenadores, ya que esta operación no admite el filtrado de etiquetas.

Para proporcionar a Tom acceso a `GetTriggers`, el administrador de AWS Glue crea una política que estructura los permisos en dos secciones. Una sección permite que Tom tenga acceso a todos los desencadenadores con la operación API `GetTriggers`. La segunda sección permite que Tom tenga acceso a las operaciones API etiquetadas con el valor `Tom`. Con esta política, Tom puede utilizar `GetTriggers` y `GetTrigger` para obtener acceso al desencadenador `t2`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "glue:GetTriggers",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "glue:*",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/Name": "Tom"
                }
            }
        }
    ]
}
```

Denegar acceso mediante etiquetas

Otra estrategia para la política de recursos consiste en denegar el acceso a los recursos explícitamente.

> **Important**
>
> Las políticas de denegación explícita no funcionan con las operaciones de API plurales, como `GetTriggers`.

En la siguiente política de ejemplo, se permiten todas las operaciones de trabajo de AWS Glue. Sin embargo, la segunda declaración `Effect` niega explícitamente el acceso a los trabajos etiquetados con la clave `Team` y el valor `Special`.

Cuando un administrador asocia la siguiente política a una identidad, esta puede acceder a todos los trabajos, excepto los que están etiquetados con la clave `Team` y el valor `Special`.

```
{
  "Version": "2012-10-17",
```

```
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "glue:*",
      "Resource": "arn:aws:glue:us-east-1:123456789012:job/*"
    },
    {
      "Effect": "Deny",
      "Action": "glue:*",
      "Resource": "arn:aws:glue:us-east-1:123456789012:job/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Team": "Special"
        }
      }
    }
  ]
}
```

Usar etiquetas con las operaciones de API List y Batch

Un tercer enfoque para elaborar una política de recursos consiste en permitir el acceso a los recursos utilizando una operación API `List` que muestre los recursos de un valor de etiqueta. A continuación, se utilizará la operación API `Batch` correspondiente, que permitirá obtener acceso a los detalles de recursos específicos. Con este enfoque, el administrador no necesita permitir el acceso las operaciones API `GetCrawlers`, `GetDevEndpoints`, `GetJobs` o `GetTriggers` plurales. En su lugar, puede proporcionar la capacidad para mostrar los recursos con las siguientes operaciones API:

- `ListCrawlers`
- `ListDevEndpoints`
- `ListJobs`
- `ListTriggers`

También puede proporcionar la posibilidad de obtener detalles sobre recursos específicos con las siguientes operaciones API:

- `BatchGetCrawlers`
- `BatchGetDevEndpoints`

- `BatchGetJobs`
- `BatchGetTriggers`

Como administrador, para utilizar este enfoque, puede realizar el siguiente procedimiento:

1. Añadir etiquetas a los rastreadores, puntos de enlace de desarrollo, trabajos y desencadenadores.
2. Denegar a los usuarios el acceso a las operaciones API `Get`, como `GetCrawlers`, `GetDevEndponts`, `GetJobs` y `GetTriggers`.
3. Para que los usuarios puedan saber a qué recursos etiquetados tienen acceso, proporcióneles acceso a las operaciones API `List`, como `ListCrawlers`, `ListDevEndponts`, `ListJobs` y `ListTriggers`.
4. Denegar a los usuarios el acceso a las API de etiquetado de AWS Glue, como `TagResource` y `UntagResource`.
5. Permitir el acceso de los usuarios a los detalles de los recursos con operaciones API `BatchGet`, como `BatchGetCrawlers`, `BatchGetDevEndponts`, `BatchGetJobs` y `BatchGetTriggers`.

Por ejemplo, cuando llame a la operación `ListCrawlers`, proporcione un valor de etiqueta que coincida con el nombre de usuario. El resultado es una lista de rastreadores que coincide con los valores de etiquetas proporcionados. Proporcione la lista de nombres a `BatchGetCrawlers` para obtener información sobre cada uno de los rastreadores con la etiqueta especificada.

Por ejemplo, si Tom solo puede recuperar información sobre los desencadenadores con la etiqueta `Tom`, el administrador puede agregar etiquetas a los desencadenadores de `Tom`, denegar a todos los usuarios el acceso a la operación de API `GetTriggers` y permitir a todos los usuarios el acceso a `ListTriggers` y `BatchGetTriggers`.

A continuación, se muestra la política de recursos que al administrador de AWS Glue concede a Tom. En la primera sección de la política, todas las operaciones API de AWS Glue están denegadas para `GetTriggers`. En la segunda sección de la política, la operación API `ListTriggers` está permitida en todos los recursos. Sin embargo, en la tercera sección, los recursos con la etiqueta `Tom` tienen permitido el acceso mediante `BatchGetTriggers`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Effect": "Deny",
            "Action": "glue:GetTriggers",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "glue:ListTriggers"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "glue:BatchGetTriggers"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/Name": "Tom"
                }
            }
        }
    ]
}
```

Si utilizamos los mismos desencadenadores que en el ejemplo anterior, Tom tendrá acceso al desencadenador `t2`, pero no al desencadenador `t1`. En el siguiente ejemplo, se muestran los resultados cuando Tom intenta obtener acceso a `t1` y `t2` con `BatchGetTriggers`.

```
aws glue batch-get-triggers --trigger-names t2
{
    "Triggers": {
        "State": "CREATED",
        "Type": "SCHEDULED",
        "Name": "t2",
        "Actions": [
            {
                "JobName": "j2"
```

```
            }
        ],
        "Schedule": "cron(0 0/1 * * ? *)"
    }
}

aws glue batch-get-triggers  --trigger-names t1

An error occurred (AccessDeniedException) when calling the BatchGetTriggers operation:
 No access to any requested resource.
```

En el siguiente ejemplo, se muestran los resultados cuando Tom intenta obtener acceso a los desencadenadores `t2` y `t3` (que no existe) en la misma llamada a `BatchGetTriggers`. Tenga en cuenta que, como Tom tiene acceso al desencadenador `t2` y dicho desencadenador existe, solo se devuelve `t2`. Aunque Tom tenga autorización para obtener acceso al desencadenador `t3`, el desencadenador `t3` no existe, por lo que, al devolver la respuesta, `t3` estará incluido en una lista de `"TriggersNotFound": []`.

```
aws glue batch-get-triggers --trigger-names t2 t3
{
    "Triggers": {
        "State": "CREATED",
        "Type": "SCHEDULED",
        "Name": "t2",
        "Actions": [
            {
                "JobName": "j2"
            }
        ],
        "TriggersNotFound": ["t3"],
        "Schedule": "cron(0 0/1 * * ? *)"
    }
}
```

Controlar la configuración mediante claves de condición o claves de contexto

Puede utilizar claves de condición o claves de contexto al conceder permisos para crear y actualizar trabajos. En estas secciones, se analizan las claves:

- Políticas que controlan la configuración mediante claves de condición
- Políticas que controlan la configuración mediante claves de contexto

Políticas que controlan la configuración mediante claves de condición

AWS Glue proporciona tres claves de condición de IAM: `glue:VpcIds`, `glue:SubnetIds` y `glue:SecurityGroupIds`. Puede utilizar claves de condición en políticas de IAM al conceder permisos para crear y actualizar trabajos. Puede utilizar esta configuración para asegurarse de que no se creen trabajos ni sesiones (ni se actualicen) para ejecutarse fuera del entorno de VPC deseado. La información de configuración de la VPC no es una entrada directa de la solicitud `CreateJob`, sino que se infiere del campo “conexiones” del trabajo que apunta a una conexión de AWS Glue.

Ejemplo de uso

Cree una conexión de tipo de red de AWS Glue denominada “traffic-monitored-connection” con el ID de la VPC deseado: “vpc-id1234”, SubnetIds y SecurityGroupIds.

Especifique la condición de las claves de condición para las acciones `CreateJob` y `UpdateJob` en la política de IAM.

```
{
  "Effect": "Allow",
  "Action": [
    "glue:CreateJob",
    "glue:UpdateJob"
  ],
  "Resource": [
    "*"
  ],
  "Condition": {
    "ForAnyValue:StringLike": {
      "glue:VpcIds": [
        "vpc-id1234"
      ]
    }
  }
}
```

Puede establecer una política de IAM similar para prohibir la creación de un trabajo de AWS Glue sin especificar información de conexión.

Restricción de sesiones en las VPC

Para hacer que las sesiones creadas se ejecuten en una VPC específica, debe restringir el permiso del rol al agregar un efecto `Deny` a la acción `glue:CreateSession` con la condición de que glue:vpc-id no sea igual a vpc-<123>. Por ejemplo:

```
"Effect": "Deny",
"Action": [
    "glue:CreateSession"
 ],
"Condition": {
    "StringNotEquals" : {"glue:VpcIds" : ["vpc-123"]}
}
```

También puede hacer que las sesiones creadas se ejecuten en una VPC al agregar un `Deny` efecto en la `glue:CreateSession` acción con la condición de que `glue:vpc-id` sea nulo. Por ejemplo:

```
{
    "Effect": "Deny",
    "Action": [
        "glue:CreateSession"
    ],
      "Condition": {
        "Null": {"glue:VpcIds": true}
    }
},
{
    "Effect": "Allow",
    "Action": [
        "glue:CreateSession"
    ],
    "Resource": ["*"]
}
```

Políticas que controlan la configuración mediante claves de contexto

AWS Glue proporciona una clave de contexto (`glue:CredentialIssuingService=glue.amazonaws.com`) a cada sesión de rol que AWS Glue pone a disposición del punto de conexión del trabajo y del desarrollador. Esto le permite implementar controles de seguridad para las acciones llevadas a cabo por scripts de AWS Glue. AWS Glue proporciona otra clave de contexto (`glue:RoleAssumedBy=glue.amazonaws.com`) a cada sesión de rol en la que AWS Glue realice

una llamada a otro servicio de AWS en nombre del cliente (no mediante un punto de conexión del trabajo o el desarrollador, sino directamente mediante el servicio de AWS Glue).

Ejemplo de uso

Especifique el permiso condicional en una política de IAM y asócielo al rol que utilizará un trabajo de AWS Glue. Esto garantiza que se permitan o denieguen determinadas acciones en función de si la sesión de rol se utiliza para el entorno de tiempo de ejecución de un trabajo de AWS Glue.

```
{
    "Effect": "Allow",
    "Action": "s3:GetObject",
    "Resource": "arn:aws:s3:::confidential-bucket/*",
    "Condition": {
        "StringEquals": {
            "glue:CredentialIssuingService": "glue.amazonaws.com"
        }
    }
}
```

Cómo denegar a una identidad la capacidad de crear sesiones de vista previa de datos

En esta sección, se incluye un ejemplo de política de IAM que se usa para denegar a una identidad la capacidad de crear sesiones de vista previa de datos. Adjunte esta política a la identidad, que es independiente del rol que se usa en la sesión de vista previa de datos durante su ejecución.

```
{
    "Sid": "DatapreviewDeny",
    "Effect": "Deny",
     "Action": [
           "glue:CreateSession"
     ],
     "Resource": [
          "arn:aws:glue:*:*:session/glue-studio-datapreview*"
      ]
 }
```

Ejemplos de políticas basadas en recursos para AWS Glue

Esta sección contiene ejemplos de políticas basadas en recursos, incluidas las políticas que conceden acceso entre cuentas.

Los ejemplos utilizan la AWS Command Line Interface (AWS CLI) para interactuar con las operaciones de API del servicio AWS Glue. Puede realizar las mismas operaciones en la consola de AWS Glue o mediante uno de los SDK de AWS.

> **Important**
>
> Al cambiar una política de recursos de AWS Glue, puede revocar accidentalmente permisos de los usuarios existentes de AWS Glue en su cuenta y causar interrupciones inesperadas. Pruebe estos ejemplos únicamente en las cuentas de desarrollo o prueba y asegúrese de que no rompen ningún flujo de trabajo existente antes de realizar los cambios.

Temas

- Consideraciones para el uso de políticas basadas en recursos con AWS Glue
- Uso de una política de recursos para controlar el acceso en la misma cuenta

Consideraciones para el uso de políticas basadas en recursos con AWS Glue

> **Note**
>
> Tanto las políticas de IAM como una política de recursos de AWS Glue tardan unos segundos en propagarse. Después de asociar una nueva política, es posible que observe que la política anterior sigue en vigor hasta que la nueva política se haya propagado por el sistema.

Puede utilizar un documento de política escrito en formato JSON para crear o modificar una política de recursos. La sintaxis de la política es la misma que para una política de IAM basada en identidades (consulte Referencia de políticas JSON de IAM), con las siguientes excepciones:

- Un bloque `"Principal"` o `"NotPrincipal"` es obligatorio para cada instrucción de política.
- `"Principal"` o `"NotPrincipal"` deben identificar a las entidades principales válidas. Los patrones de comodín (como `arn:aws:iam::`*`account-id`*`:user/*`) no están permitidos.
- El bloque `"Resource"` en la política requiere que todos los ARN de recursos se correspondan con la siguiente sintaxis de expresiones regulares (donde el primer `%s` es la *`región`* y el segundo `%s` es el *`ID de cuenta`*):

```
*arn:aws:glue:%s:%s:(\*|[a-zA-Z\*]+\/?.*)
```

Por ejemplo, `arn:aws:glue:us-west-2:`*`account-id`*`:*` y `arn:aws:glue:us-west-2:`*`account-id`*`:database/default` están permitidos, pero `*` no está permitido.

- A diferencia de las políticas basadas en identidades, una política de recursos de AWS Glue solo debe contener nombres de recursos de Amazon (ARN) que pertenecen al catálogo al que se asocia dicha política. Estos ARN siempre empiezan por `arn:aws:glue:`.
- Una política no puede impedir que la identidad que la cree genere o modifique otras políticas.
- Un documento JSON de política de recursos no puede superar los 10 KB de tamaño.

Uso de una política de recursos para controlar el acceso en la misma cuenta

En este ejemplo, un usuario administrador en la cuenta A crea una política de recursos que concede al usuario de IAM, `Alice`, de la cuenta A un acceso completo al catálogo. Alice no tiene ninguna política de IAM asociada.

Para ello, el usuario administrador ejecuta el siguiente comando de la AWS CLI.

```
# Run as admin of Account A
$ aws glue put-resource-policy --profile administrator-name --region us-west-2 --
policy-in-json '{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Principal": {
        "AWS": [
          "arn:aws:iam::account-A-id:user/Alice"
        ]
      },
      "Effect": "Allow",
      "Action": [
        "glue:*"
      ],
      "Resource": [
        "arn:aws:glue:us-west-2:account-A-id:*"
      ]
    }
  ]
}'
```

En lugar de especificar el documento de política JSON como parte de su comando de AWS CLI, puede guardar un documento de política en un archivo y hacer referencia a la ruta del archivo en el comando de AWS CLI, con el prefijo `file://`. A continuación, se muestra un ejemplo de cómo podría hacerlo.

```
$ echo '{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Principal": {
        "AWS": [
          "arn:aws:iam::account-A-id:user/Alice"
        ]
      },
      "Effect": "Allow",
      "Action": [
        "glue:*"
      ],
      "Resource": [
        "arn:aws:glue:us-west-2:account-A-id:*"
      ]
    }
  ]
}' > /temp/policy.json

$ aws glue put-resource-policy --profile admin1 \
    --region us-west-2 --policy-in-json file:///temp/policy.json
```

Después de que esta política de recurso se haya propagado, Alice puede acceder a todos los recursos de AWS Glue de la cuenta A; de la siguiente manera.

```
# Run as user Alice
$ aws glue create-database --profile alice --region us-west-2 --database-input '{
    "Name": "new_database",
    "Description": "A new database created by Alice",
    "LocationUri": "s3://my-bucket"
}'

$ aws glue get-table --profile alice --region us-west-2 --database-name "default" --
table-name "tbl1"}
```

En respuesta a la llamada `get-table` de Alice, el servicio de AWS Glue devuelve lo siguiente.

```
{
  "Table": {
    "Name": "tbl1",
    "PartitionKeys": [],
    "StorageDescriptor": {
        ......
    },
    ......
  }
}
```

AWS políticas gestionadas para AWS Glue

Una política AWS gestionada es una política independiente creada y administrada por AWS. AWS Las políticas administradas están diseñadas para proporcionar permisos para muchos casos de uso comunes, de modo que pueda empezar a asignar permisos a usuarios, grupos y funciones.

Ten en cuenta que es posible que las políticas AWS administradas no otorguen permisos con privilegios mínimos para tus casos de uso específicos, ya que están disponibles para que los usen todos los AWS clientes. Se recomienda definir políticas administradas por el cliente específicas para sus casos de uso a fin de reducir aún más los permisos.

No puedes cambiar los permisos definidos en AWS las políticas administradas. Si AWS actualiza los permisos definidos en una política AWS administrada, la actualización afecta a todas las identidades principales (usuarios, grupos y roles) a las que está asociada la política. AWS es más probable que actualice una política AWS administrada cuando Servicio de AWS se lance una nueva o cuando estén disponibles nuevas operaciones de API para los servicios existentes.

Para obtener más información, consulte Políticas administradas por AWS en la Guía del usuario de IAM.

AWS políticas gestionadas (predefinidas) para AWS Glue

AWS aborda muchos casos de uso comunes al proporcionar políticas de IAM independientes que son creadas y administradas por. AWS Estas políticas AWS gestionadas conceden los permisos necesarios para los casos de uso más habituales, de forma que no tenga que investigar qué permisos son necesarios. Para obtener más información, consulte Políticas administradas por AWS en la Guía del usuario de IAM.

Las siguientes políticas AWS gestionadas, que puedes adjuntar a las identidades de tu cuenta, son específicas AWS Glue y están agrupadas por escenarios de uso:

- AWSGlueConsoleFullAccess— Otorga acceso total a AWS Glue los recursos cuando una identidad a la que está asociada la política utiliza la AWS Management Console. Si sigue la convención de nomenclatura para los recursos especificados en esta política, los usuarios dispondrán de todas las funciones de la consola. Esta política se suele adjuntar a los usuarios de la consola AWS Glue.
- AWSGlueServiceRole— Otorga acceso a los recursos que varios AWS Glue procesos requieren para ejecutarse en su nombre. Estos recursos incluyen AWS Glue Amazon S3, IAM, CloudWatch Logs y Amazon EC2. Si aplica la convención de nomenclatura en los recursos especificados en esta política, los procesos de AWS Glue tienen los permisos necesarios. Normalmente, esta política se asocia a los roles que se especifican a la hora de definir rastreadores, trabajos y puntos de conexión de desarrollo.
- AwsGlueSessionUserRestrictedServiceRole— Proporciona acceso completo a todos los AWS Glue recursos, excepto a las sesiones. Permite a los usuarios crear y utilizar solo las sesiones interactivas que están asociadas al usuario. Esta política incluye otros permisos necesarios AWS Glue para administrar AWS Glue los recursos en otros AWS servicios. La política también permite añadir etiquetas a AWS Glue los recursos de otros AWS servicios.

  > **Note**
  >
  > Para obtener los beneficios de seguridad completos, no conceda esta política a un usuario al que se le haya asignado la política `AWSGlueServiceRole`, `AWSGlueConsoleFullAccess` o `AWSGlueConsoleSageMakerNotebookFullAccess`.

- AwsGlueSessionUserRestrictedPolicy— Proporciona acceso para crear sesiones AWS Glue interactivas mediante la operación de la `CreateSession` API solo si se proporcionan una clave de etiqueta («propietario» y un valor que coincidan con el ID de AWS usuario del cesionario). Esta política de identidad está asociada al usuario de IAM que invoca a la operación de la API `CreateSession`. Esta política también permite al cesionario interactuar con los recursos de la sesión AWS Glue interactiva que se crearon con una etiqueta de «propietario» y un valor que coincidan con su ID de usuario. AWS Esta política deniega el permiso para cambiar o eliminar las etiquetas de “propietario” de un recurso de sesión de AWS Glue luego de que se crea la sesión.

> **Note**
>
> Para obtener los beneficios de seguridad completos, no conceda esta política a un usuario al que se le haya asignado la política `AWSGlueServiceRole`, `AWSGlueConsoleFullAccess` o `AWSGlueConsoleSageMakerNotebookFullAccess`.

- AwsGlueSessionUserRestrictedNotebookServiceRole— Proporciona acceso suficiente a la sesión del AWS Glue Studio bloc de notas para interactuar con recursos específicos de la sesión AWS Glue interactiva. Se trata de recursos que se crean con el valor de la etiqueta «propietario» que coincide con el ID de AWS usuario del principal (usuario o rol de IAM) que crea el bloc de notas. Para obtener más información sobre estas etiquetas, consulte el gráfico Valores clave de la entidad principal en la Guía del usuario de IAM.

  Esta política de rol de servicio se asocia al rol que se especifica con un comando mágico dentro del cuaderno o que se transfiere como un rol a la operación de la API `CreateSession`. Esta política también permite al director crear una sesión AWS Glue interactiva desde la interfaz del AWS Glue Studio bloc de notas solo si la clave de la etiqueta «propietario» y el valor coinciden con el ID de AWS usuario del director. Esta política deniega el permiso para cambiar o eliminar las etiquetas de “propietario” de un recurso de sesión de AWS Glue luego de que se crea la sesión. Esta política también incluye permisos para escribir y leer desde buckets de Amazon S3, escribir CloudWatch registros y crear y eliminar etiquetas para los recursos de Amazon EC2 utilizados por. AWS Glue

  > **Note**
  >
  > Para obtener todos los beneficios de seguridad, no conceda esta política a un rol al que se le haya asignado la política `AWSGlueServiceRole`, `AWSGlueConsoleFullAccess` o `AWSGlueConsoleSageMakerNotebookFullAccess`.

- AwsGlueSessionUserRestrictedNotebookPolicy— Solo permite crear una sesión AWS Glue interactiva desde la interfaz del AWS Glue Studio bloc de notas si hay una clave de etiqueta («propietario») y un valor que coincidan con el identificador de AWS usuario del principal (usuario o rol de IAM) que crea el bloc de notas. Para obtener más información sobre estas etiquetas, consulte el gráfico Valores clave de la entidad principal en la Guía del usuario de IAM.

  Esta política se adjunta a la entidad principal (usuario o rol de IAM) que crea sesiones a partir de la interfaz del cuaderno de AWS Glue Studio. Esta política también permite un acceso suficiente

al cuaderno de AWS Glue Studio para interactuar con recursos de sesión interactivos de AWS Glue específicos. Se trata de recursos que se crean con el valor de la etiqueta «propietario» que coincide con el ID de AWS usuario del director. Esta política deniega el permiso para cambiar o eliminar las etiquetas de “propietario” de un recurso de sesión de AWS Glue luego de que se crea la sesión.

- AWSGlueServiceNotebookRole— Otorga acceso a AWS Glue las sesiones iniciadas en un AWS Glue Studio cuaderno. Esta política permite enumerar y obtener la información de todas las sesiones, pero solo permite a los usuarios crear y usar las sesiones etiquetadas con su AWS seudónimo. Esta política deniega el permiso para cambiar o eliminar las etiquetas de «propietario» de los recursos de AWS Glue sesión etiquetadas con su AWS ID.

  Asigne esta política al AWS usuario que crea trabajos mediante la interfaz del bloc de notas deAWS Glue Studio.
- AWSGlueConsoleSageMakerNotebookFullAccess— Otorga acceso completo a AWS Glue y a SageMaker los recursos cuando la identidad a la que se adjunta la política utiliza el AWS Management Console. Si sigue la convención de nomenclatura para los recursos especificados en esta política, los usuarios dispondrán de todas las funciones de la consola. Esta política suele estar asociada a los usuarios de la AWS Glue consola que gestionan SageMaker ordenadores portátiles.
- AWSGlueSchemaRegistryFullAccess— Concede acceso total a los recursos de AWS Glue Schema Registry cuando la identidad a la que está asociada la política utiliza la tecla AWS Management Console o AWS CLI. Si sigue la convención de nomenclatura para los recursos especificados en esta política, los usuarios dispondrán de todas las funciones de la consola. Por lo general, esta política se adjunta a los usuarios de la AWS Glue consola o AWS CLI que administran el registro de AWS Glue esquemas.
- AWSGlueSchemaRegistryReadonlyAccess— Otorga acceso de solo lectura a los recursos del Registro de AWS Glue esquemas cuando una identidad a la que está asociada la política utiliza la AWS Management Console tecla o. AWS CLI Si sigue la convención de nomenclatura para los recursos especificados en esta política, los usuarios dispondrán de todas las funciones de la consola. Esta política suele estar asociada a los usuarios de la AWS Glue consola o AWS CLI que utilizan el registro de AWS Glue esquemas.

> **Note**
>
> Para consultar estas políticas de permisos, inicie sesión en la consola de IAM y busque las políticas específicas.

También puede crear sus propias políticas de IAM personalizadas para conceder permisos a las acciones y recursos de AWS Glue. Puede asociar estas políticas personalizadas a los grupos o usuarios de IAM que requieran esos permisos.

AWS Glue actualiza las políticas AWS gestionadas

Consulta los detalles sobre las actualizaciones de las políticas AWS gestionadas de AWS Glue desde que este servicio comenzó a realizar el seguimiento de estos cambios. Para recibir alertas automáticas sobre los cambios en esta página, suscríbete a la fuente RSS de la página de historial de documentos de AWS Glue.

Cambio	Descripción	Fecha
AwsGlueSessionUserRestrictedPolicy — Actualización menor de una política existente.	Agregar `glue:StartCompletion` y `glue:GetCompletion` a la política. Necesario para la integración de datos de Amazon Q en AWS Glue.	30 de abril de 2024
AwsGlueSessionUserRestrictedNotebookServiceRole — Actualización menor de una política existente.	Agregar `glue:StartCompletion` y `glue:GetCompletion` a la política. Necesario para la integración de datos de Amazon Q en AWS Glue.	30 de abril de 2024
AwsGlueSessionUserRestrictedServiceRole — Actualización menor de una política existente.	Agregar `glue:StartCompletion` y `glue:GetCompletion` a la política. Necesario para la integración de datos de Amazon Q en AWS Glue.	30 de abril de 2024
AWSGlueServiceNotebookRole — Actualiza	Agregar `glue:StartCompletion` y `glue:GetCompletion`	30 de enero de 2024

Cambio	Descripción	Fecha
ción menor de una política existente.	a la política. Necesario para la integración de datos de Amazon Q en AWS Glue.	
AwsGlueSessionUserRestrictedNotebookPolicy — Actualización menor de una política existente.	Agregar `glue:StartCompletion` y `glue:GetCompletion` a la política. Necesario para la integración de datos de Amazon Q en AWS Glue.	29 de noviembre de 2023
AWSGlueServiceNotebookRole — Actualización menor de una política existente.	Agregar `codewhisperer:GenerateRecommendations` a la política. Necesario para una nueva función en la que AWS Glue genera CodeWhisperer recomendaciones.	9 de octubre de 2023
AWSGlueServiceRole — Actualización menor de una política existente.	Reduzca el alcance de CloudWatch los permisos para reflejar mejor el registro de AWS Glue.	4 de agosto de 2023
AWSGlueConsoleFullAccess — Actualización menor de una política existente.	Agregar una lista de recetas de `databrew` y describir los permisos a la política. Necesario para proporcionar acceso administrativo completo a las nuevas funciones en las que AWS Glue puede acceder a las recetas.	9 de mayo de 2023

Cambio	Descripción	Fecha
AWSGlueConsoleFullAccess — Actualización menor de una política existente.	Agregar `cloudformation:ListStacks` a la política. Conserva las capacidades existentes tras los cambios en los requisitos de AWS CloudFormation autorización.	28 de marzo de 2023
Se agregaron nuevas políticas administradas para la característica de sesiones interactivas: • AwsGlueSessionUserRestrictedServiceRole • AwsGlueSessionUserRestrictedPolicy • AwsGlueSessionUserRestrictedNotebookServiceRole • AwsGlueSessionUserRestrictedNotebookPolicy	Estas políticas se diseñaron para proporcionar seguridad adicional para las sesiones interactivas y los cuadernos en AWS Glue Studio. Las políticas restringen el acceso a la operación de la API `CreateSession` para que solo el propietario tenga acceso.	30 de noviembre de 2021

Cambio	Descripción	Fecha
AWSGlueConsoleSageMakerNotebookFullAccess — Actualización de una política existente.	Se eliminó un ARN de recurso redundante (`arn:aws:s3:::aws-glue-*/*`) para la acción que concede permisos de lectura/escritura en los buckets de Amazon S3 que AWS Glue utiliza para almacenar scripts y archivos temporales. Se corrigió un problema de sintaxis al cambiar `"StringEquals"` a `"ForAnyValue:StringLike"`, y se movieron las líneas `"Effect": "Allow"` para que precedan a la línea `"Action":` en las instancias en la que no funcionaban.	15 de julio de 2021
AWSGlueConsoleFullAccess — Actualización de una política existente.	Se eliminó un ARN de recurso redundante (`arn:aws:s3:::aws-glue-*/*`) para la acción que concede permisos de lectura/escritura en los buckets de Amazon S3 que AWS Glue utiliza para almacenar scripts y archivos temporales.	15 de julio de 2021
AWS Glue comenzó el seguimiento de los cambios.	AWS Gluecomenzó a realizar un seguimiento de los cambios en sus políticas AWS gestionadas.	10 de junio de 2021

Especificación de ARN del recurso de AWS Glue

En AWS Glue, puede controlar el acceso a los recursos a través de una política de AWS Identity and Access Management (IAM). En las políticas se emplean nombres de recurso de Amazon (ARN) para identificar los recursos a los que se aplican las políticas. No todos los recursos de AWS Glue admiten ARN.

Temas

- ARN de Data Catalog
- ARN para objetos que no están en el catálogo en AWS Glue
- Control de acceso para operaciones API singulares que no estén en el catálogo de AWS Glue
- Control de acceso para operaciones API que no están en el catálogo de AWS Glue que recuperan varios elementos
- Control de acceso para operaciones de API BatchGet que no están en el catálogo de AWS Glue

ARN de Data Catalog

Los recursos de Data Catalog tienen una estructura jerárquica, con `catalog` como la raíz.

```
arn:aws:glue:region:account-id:catalog
```

Cada cuenta de AWS tiene un único Data Catalog en una región de AWS con el ID de cuenta de 12 dígitos como el ID de catálogo. Los recursos tienen ARN únicos asociados a ellos, tal y como se muestra en la siguiente tabla.

Tipo de recurso	Formato de ARN
Catálogo	`arn:aws:glue:` *`region`*`:`*`account-id`* `:catalog` Por ejemplo: `arn:aws:glue:us-east-1:123456789012:catalog` .
Base de datos	`arn:aws:glue:` *`region`*`:`*`account-id`* `:database/` *`database name`* Por ejemplo: `arn:aws:glue:us-east-1:123456789012:database/db1` .

Tipo de recurso	Formato de ARN
Tabla	`arn:aws:glue:` *`region`*`:`*`account-id`* `:table/`*`database name`*`/`*`table name`* Por ejemplo: `arn:aws:glue:us-east-1:123456789012:table/db1/tbl1` .
Función definida por el usuario	`arn:aws:glue:` *`region`*`:`*`account-id`* `:userDefinedFunction/` *`database name`*`/`*`user-defined function name`* Por ejemplo: `arn:aws:glue:us-east-1:123456789012:userDefinedFunction/db1/func1` .
Connection	`arn:aws:glue:` *`region`*`:`*`account-id`* `:connection/` *`connection name`* Por ejemplo: `arn:aws:glue:us-east-1:123456789012:connection/connection1` .
Sesiones interactivas	`arn:aws:glue:` *`region`*`:`*`account-id`* `:session/` *`interactive session id`* Por ejemplo: `arn:aws:glue:us-east-1:123456789012:session/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111` .

Para habilitar el control de acceso detallado, puede utilizar estos ARN en sus políticas de IAM y las políticas de recursos para conceder y denegar el acceso a recursos específicos. Los comodines están permitidos en las políticas. Por ejemplo, el siguiente ARN coincide con todas las tablas de la base de datos `default`.

```
arn:aws:glue:us-east-1:123456789012:table/default/*
```

> **Important**
>
> Todas las operaciones realizadas en un recurso de Data Catalog requieren permiso en el recurso y todos los antecesores de dicho recurso. Por ejemplo, para crear una partición de una tabla se requiere el permiso en la tabla, la base de datos y el catálogo donde se

encuentra la tabla. El siguiente ejemplo muestra el permiso necesario para crear particiones en la tabla `PrivateTable` en la base de datos `PrivateDatabase` de Data Catalog.

```
{
   "Sid": "GrantCreatePartitions",
   "Effect": "Allow",
   "Action": [
       "glue:BatchCreatePartitions"
   ],
   "Resource": [
       "arn:aws:glue:us-east-1:123456789012:table/PrivateDatabase/PrivateTable",
       "arn:aws:glue:us-east-1:123456789012:database/PrivateDatabase",
       "arn:aws:glue:us-east-1:123456789012:catalog"
   ]
}
```

Además de permiso en el recurso y todos sus antecesores, todas las operaciones de eliminación requieren permiso en todos los elementos secundarios de dicho recurso. Por ejemplo, para eliminar una base de datos, se requiere permiso en todas las tablas y funciones definidas por el usuario en la base de datos, así como en la base de datos y el catálogo donde se encuentra la base de datos. El siguiente ejemplo muestra el permiso necesario para eliminar la base de datos `PrivateDatabase` del Data Catalog.

```
{
   "Sid": "GrantDeleteDatabase",
   "Effect": "Allow",
   "Action": [
       "glue:DeleteDatabase"
   ],
   "Resource": [
       "arn:aws:glue:us-east-1:123456789012:table/PrivateDatabase/*",
       "arn:aws:glue:us-east-1:123456789012:userDefinedFunction/PrivateDatabase/
*",
       "arn:aws:glue:us-east-1:123456789012:database/PrivateDatabase",
       "arn:aws:glue:us-east-1:123456789012:catalog"
   ]
}
```

En resumen, las acciones en los recursos de Data Catalog siguen estas reglas de permisos:

- Las acciones en el catálogo requieren permiso solo en el catálogo.

- Las acciones en una base de datos requieren permiso en la base de datos y el catálogo.
- Eliminar acciones en una base de datos requiere permiso en la base de datos y el catálogo, además de todas las tablas y funciones definidas por el usuario en la base de datos.
- Las acciones en una tabla, partición o versión de tabla requieren el permiso en la tabla, base de datos y catálogo.
- Las acciones en una función definida por el usuario requieren el permiso en la función definida por el usuario, la base de datos y el catálogo.
- Las acciones en una conexión requieren permiso en la conexión y el catálogo.

ARN para objetos que no están en el catálogo en AWS Glue

Algunos recursos de AWS Glue permiten permisos de nivel de recursos para controlar el acceso mediante un ARN. Puede utilizar estos ARN en sus políticas de IAM para habilitar el control de acceso detallado. En la siguiente tabla se enumeran los recursos que puede contener ARN de recursos.

Tipo de recurso	Formato de ARN
Rastreador	`arn:aws:glue:` *`region`*`:`*`account-id`* `:crawler/` *`crawler-name`* Por ejemplo: `arn:aws:glue:us-east-1:123456789012:crawler/mycrawler` .
Trabajo	`arn:aws:glue:` *`region`*`:`*`account-id`* `:job/`*`job-name`* Por ejemplo: `arn:aws:glue:us-east-1:123456789012:job/testjob` .
Desencadenador	`arn:aws:glue:` *`region`*`:`*`account-id`* `:trigger/` *`trigger-name`* Por ejemplo: `arn:aws:glue:us-east-1:123456789012:trigger/sampletrigger` .

Tipo de recurso	Formato de ARN
Punto de enlace de desarrollo	`arn:aws:glue:` *`region`*`:`*`account-id`* `:devEndpoint/` *`development-endpoint-name`* Por ejemplo: `arn:aws:glue:us-east-1:123456789012:devEndpoint/temporarydevendpoint` .
Transformación de machine learning	`arn:aws:glue:` *`region`*`:`*`account-id`* `:mlTransform/` *`transform-id`* Por ejemplo: `arn:aws:glue:us-east-1:123456789012:mlTransform/tfm-1234567890` .

Control de acceso para operaciones API singulares que no estén en el catálogo de AWS Glue

Las operaciones de API singulares que no estén en el catálogo de AWS Glue actúan sobre un solo elemento (punto de conexión de desarrollo). Algunos ejemplos son `GetDevEndpoint`, `CreateUpdateDevEndpoint` y `UpdateDevEndpoint`. Para estas operaciones, una política debe poner el nombre de la API en el bloque `"action"` y el ARN del recurso en el bloque `"resource"`.

Supongamos que desea permitir a un usuario llamar a la operación `GetDevEndpoint`. La siguiente política concede los permisos mínimos necesarios a un punto de enlace llamado `myDevEndpoint-1`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "MinimumPermissions",
            "Effect": "Allow",
            "Action": "glue:GetDevEndpoint",
            "Resource": "arn:aws:glue:us-east-1:123456789012:devEndpoint/
myDevEndpoint-1"
        }
    ]
}
```

La siguiente política permite a `UpdateDevEndpoint` el acceso a los recursos que coinciden con `myDevEndpoint-` con un asterisco (*).

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PermissionWithWildcard",
            "Effect": "Allow",
            "Action": "glue:UpdateDevEndpoint",
            "Resource": "arn:aws:glue:us-east-1:123456789012:devEndpoint/myDevEndpoint-
*"
        }
    ]
}
```

Puede combinar los dos políticas tal y como se muestra en el siguiente ejemplo. Puede que vea `EntityNotFoundException` para cualquier punto de enlace de desarrollo cuyo nombre empiece por `A`. Sin embargo, se devuelve un error de acceso denegado cuando intenta acceder a otros puntos de enlace de desarrollo.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CombinedPermissions",
            "Effect": "Allow",
            "Action": [
                "glue:UpdateDevEndpoint",
                "glue:GetDevEndpoint"
            ],
            "Resource": "arn:aws:glue:us-east-1:123456789012:devEndpoint/A*"
        }
    ]
}
```

Control de acceso para operaciones API que no están en el catálogo de AWS Glue que recuperan varios elementos

Algunas operaciones API de AWS Glue recupera varios elementos (como varios puntos de enlace de desarrollo); por ejemplo, `GetDevEndpoints`. Para esta operación, solo puede especificar un recurso de asterisco (*) y no ARN específicos.

Por ejemplo, para incluir `GetDevEndpoints` en la política, el ámbito del recurso debe establecer en el carácter comodín (*). Las operaciones singulares (`GetDevEndpoint`, `CreateDevEndpoint` y `DeleteDevendpoint`) también se asignan a todos los recursos (*) en el ejemplo.

```
{
            "Sid": "PluralAPIIncluded",
            "Effect": "Allow",
            "Action": [
                "glue:GetDevEndpoints",
                "glue:GetDevEndpoint",
                "glue:CreateDevEndpoint",
                "glue:UpdateDevEndpoint"
            ],
            "Resource": [
                "*"
            ]
}
```

Control de acceso para operaciones de API BatchGet que no están en el catálogo de AWS Glue

Algunas operaciones API de AWS Glue recupera varios elementos (como varios puntos de enlace de desarrollo); por ejemplo, `BatchGetDevEndpoints`. En esta operación, puede especificar un ARN para limitar el ámbito de los recursos a los que se puede obtener acceso.

Por ejemplo, para permitir el acceso a un punto de enlace de desarrollo específico, incluya `BatchGetDevEndpoints` en la política con el ARN del recurso.

```
{
            "Sid": "BatchGetAPIIncluded",
            "Effect": "Allow",
            "Action": [
                "glue:BatchGetDevEndpoints"
```

```
            ],
            "Resource": [
                "arn:aws:glue:us-east-1:123456789012:devEndpoint/de1"
            ]
}
```

Con esta política, puede obtener acceso correctamente al punto de enlace de desarrollo llamado `de1`. Sin embargo, si intenta obtener acceso al punto de enlace de desarrollo llamado `de2`, se devolverá un error.

```
An error occurred (AccessDeniedException) when calling the BatchGetDevEndpoints
 operation: No access to any requested resource.
```

> **Important**
>
> Si desea conocer otros enfoques alternativos para configurar las políticas de IAM, como el uso de las operaciones API `List` y `BatchGet`, consulte Ejemplos de políticas basadas en identidades para AWS Glue.

Concesión de acceso entre cuentas

Al conceder acceso a recursos de Data Catalog entre cuentas, los trabajos de extracción, transformación y carga (ETL) pueden consultar y combinar datos de diferentes cuentas.

Temas

- Métodos para conceder acceso entre cuentas en AWS Glue
- Agregar o actualizar la política de recursos de Data Catalog
- Realización de una llamada a la API entre cuentas
- Realización de una llamada a ETL entre cuentas
- Registro entre cuentas de CloudTrail
- Propiedad de recursos entre cuentas y facturación
- Limitaciones de acceso entre cuentas

Métodos para conceder acceso entre cuentas en AWS Glue

Puede conceder acceso a sus datos a cuentas de AWS externas mediante los métodos de AWS Glue o mediante el uso de concesiones entre cuentas de AWS Lake Formation. Los métodos de AWS Glue utilizan políticas de AWS Identity and Access Management (IAM) para lograr un control preciso de acceso. Lake Formation utiliza un modelo de permisos `GRANT/REVOKE` similares a los comandos de `GRANT/REVOKE` en un sistema de base de datos relacional.

En esta sección, se describe el uso de los métodos de AWS Glue. Para obtener más información sobre cómo utilizar concesiones entre cuentas de Lake Formation, consulte Concesión de permisos de Lake Formation en la Guía para desarrolladores de AWS Lake Formation.

Hay dos métodos de AWS Glue para conceder acceso entre cuentas a un recurso:

- Uso de una política de recursos de Data Catalog
- Uso de un rol de IAM

Concesión de acceso entre cuentas con una política de recursos

Los siguientes son los pasos generales para conceder acceso entre cuentas mediante una política de recursos de Data Catalog:

1. Un administrador (u otra identidad autorizada) en la cuenta A asocia una política de recursos a Data Catalog en la cuenta A. Esta política concede a la cuenta B permisos específicos entre cuentas para realizar operaciones en un recurso en el catálogo de la cuenta A.
2. Un administrador de la cuenta B asocia una política de IAM a una identidad de IAM en la cuenta B que delega los permisos recibidos de la cuenta A.

   La identidad de la cuenta B ahora tiene acceso al recurso especificado en la cuenta A.

   La identidad necesita permiso de ambos, el propietario de los recursos (cuenta A) y su cuenta principal (cuenta B), para poder obtener acceso al recurso.

Conceder acceso entre cuentas mediante un rol de IAM

Los siguientes son los pasos generales para conceder acceso entre cuentas mediante un rol de IAM:

1. Un administrador (u otra identidad autorizada) en la cuenta que posee el recurso (cuenta A) crea un rol de IAM.

2. El administrador de la cuenta A asocia una política al rol que concede permisos entre cuentas para acceder al recurso en cuestión.
3. El administrador de la cuenta A asocia al rol una política de confianza que identifica una identidad de IAM en una cuenta diferente (cuenta B) como la entidad principal que puede asumir el rol.

   La entidad principal de la política de confianza también puede ser una entidad principal de un servicio de AWS, si desea conceder permisos a un servicio de AWS para que asuma el rol.
4. Un administrador de la cuenta B ahora delega los permisos a una o varias identidades de IAM de la cuenta B para que puedan asumir ese rol. De esta forma, ofrece a las identidades de la cuenta B acceso al recurso de la cuenta A.

Para obtener más información sobre el uso de IAM para delegar permisos, consulte Access management (Administración de accesos) en la Guía del usuario de IAM. Para obtener más información acerca de los usuarios, los grupos, los roles y los permisos, consulte Identidades (usuarios, grupos y roles) en la Guía del usuario de IAM.

Si desea obtener una comparación de estos dos enfoques, consulte Cómo los roles de IAM difieren de las políticas basadas en recursos en la Guía del usuario de IAM. AWS Glue es compatible con ambas opciones, con la restricción de que una política de recursos puede conceder acceso únicamente a los recursos de Catálogo de datos.

Por ejemplo, para conceder al rol `Dev` de la cuenta B acceso a la base de datos `db1` en la cuenta A, asocie la siguiente política de recursos al catálogo de la cuenta A.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "glue:GetDatabase"
      ],
      "Principal": {"AWS": [
        "arn:aws:iam::account-B-id:role/Dev"
      ]},
      "Resource": [
        "arn:aws:glue:us-east-1:account-A-id:catalog",
        "arn:aws:glue:us-east-1:account-A-id:database/db1"
      ]
    }
```

```
  ]
}
```

Además, la cuenta B tendría que asociar la siguiente política de IAM al rol `Dev` para obtener acceso a db1 en la cuenta A.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "glue:GetDatabase"
      ],
      "Resource": [
        "arn:aws:glue:us-east-1:account-A-id:catalog",
        "arn:aws:glue:us-east-1:account-A-id:database/db1"
      ]
    }
  ]
}
```

Agregar o actualizar la política de recursos de Data Catalog

Puede agregar o actualizar la política de recursos de Data Catalog de AWS Glue mediante la consola, la API o AWS Command Line Interface (AWS CLI).

> **Important**
>
> Si ya ha otorgado permisos entre cuentas desde su cuenta con AWS Lake Formation, agregar o actualizar la política de recursos de Data Catalog requiere un paso adicional. Para obtener más información, consulte Administrar los permisos entre cuentas utilizando AWS Glue y Lake Formation en la Guía para desarrolladores de AWS Lake Formation.
> Para determinar si existen concesiones entre cuentas de Lake Formation, utilice la operación de API `glue:GetResourcePolicies` o la AWS CLI. Si `glue:GetResourcePolicies` devuelve una política que no sea de Catálogo de datos, se cuenta con concesiones de Lake Formation. Para obtener más información, consulte Visualización de todas las concesiones entre cuentas mediante la operación de API GetResourcePolicies en la Guía para desarrolladores de AWS Lake Formation.

Agregar o actualizar la política de recursos de Data Catalog (consola)

1. Abra la consola de AWS Glue en https://console.aws.amazon.com/glue/.

   Inicie sesión como usuario administrativo de AWS Identity and Access Management (IAM) con permiso `glue:PutResourcePolicy`.
2. En el panel de navegación, seleccione Configuración.
3. En la página Data catalog settings (Configuración de Data Catalog), en Permissions (Permisos), pegue una política de recursos en el área de texto. A continuación, elija Guardar.

   Si la consola muestra una alerta que indica que los permisos de la política se agregarán a los permisos concedidos mediante Lake Formation, elija Proceed (Proceder).

Agregar o actualizar la política de recursos de Data Catalog (AWS CLI)

- Envíe un comando `aws glue put-resource-policy`. Si ya existen concesiones de Lake Formation, asegúrese de incluir la opción `--enable-hybrid` con el valor `'TRUE'`.

  Para obtener ejemplos de uso de este comando, consulte Ejemplos de políticas basadas en recursos para AWS Glue.

Realización de una llamada a la API entre cuentas

Todas las operaciones de AWS Glue Data Catalog tienen un campo `CatalogId`. Si se han concedido los permisos necesarios para permitir el acceso entre cuentas, un intermediario puede realizar llamadas a la API de Data Catalog entre cuentas. El intermediario realiza este procedimiento pasando el ID de cuenta de AWS de destino en `CatalogId` de manera que acceda al recurso en esa cuenta de destino.

Si no se proporciona ningún valor `CatalogId`, AWS Glue utiliza el propio ID de cuenta del intermediario de forma predeterminada y la llamada no será entre cuentas.

Realización de una llamada a ETL entre cuentas

Algunas API de AWS Glue PySpark y Scala tienen un campo de ID de catálogo. Si se han concedido todos los permisos necesarios para habilitar el acceso entre cuentas, un trabajo de ETL puede realizar llamadas de PySpark y Scala a las operaciones de la API entre cuentas, mediante la transferencia del ID de la cuenta de AWS de destino en el campo de ID de catálogo, con el fin de obtener acceso a los recursos de Data Catalog de una cuenta de destino.

Si no se proporciona ningún valor de ID de catálogo, AWS Glue utiliza el propio ID de cuenta del intermediario de forma predeterminada y la llamada no será entre cuentas.

Para API de PySpark que admiten `catalog_id`, consulte GlueContext clase. Para API de Scala que admiten `catalogId`, consulte AWS GlueAPI de Scala GlueContext .

El siguiente ejemplo muestra los permisos que requiere al beneficiario para ejecutar un trabajo de ETL. En este ejemplo, *`grantee-account-id`* es el `catalog-id` del cliente que ejecuta el trabajo y *`grantor-account-id`* es el propietario del recurso. En este ejemplo se concede permiso a todos los recursos de catálogo de la cuenta del concedente. Para limitar el alcance de recursos concedidos, puede proporcionar ARN específicas para el catálogo, la base de datos, la tabla y la conexión.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "glue:GetConnection",
                "glue:GetDatabase",
                "glue:GetTable",
                "glue:GetPartition"
            ],
            "Principal": {"AWS": ["arn:aws:iam::grantee-account-id:root"]},
            "Resource": [
                "arn:aws:glue:us-east-1:grantor-account-id:*"
            ]
        }
    ]
}
```

> **Note**
>
> Si una tabla de la cuenta del concedente apunta a una ubicación de Amazon S3, es decir, también en la cuenta del concedente, el rol de IAM que se utiliza para ejecutar un trabajo de ETL en la cuenta del beneficiario debe tener permiso para enumerar y obtener los objetos de la cuenta del concedente.

Dado que el cliente de la cuenta A ya tiene permiso para crear y ejecutar trabajos de ETL, a continuación se muestran los pasos básicos para configurar un trabajo de ETL para el acceso entre cuentas:

1. Permita el acceso a los datos entre cuentas (omitir este paso si el acceso entre cuentas de Amazon S3 ya está configurado).
    a. Actualice la política de bucket de Amazon S3 en la cuenta B para permitir el acceso entre cuentas desde la cuenta A.
    b. Actualizar la política de IAM de la cuenta A para permitir el acceso al bucket en la cuenta B.
2. Permita acceso entre cuentas al Data Catalog.
    a. Cree o actualice la política de recursos asociada al Data Catalog en la cuenta B para permitir el acceso desde la cuenta A.
    b. Actualice la política de IAM de la cuenta A para permitir el acceso a Data Catalog en la cuenta B.

Registro entre cuentas de CloudTrail

Cuando un trabajo del servicio ETL (extracción, transformación y carga) de AWS Glue tiene acceso a los datos subyacentes de una tabla de Data Catalog compartida mediante concesiones entre cuentas de AWS Lake Formation, existe una conducta de registro adicional de AWS CloudTrail.

A los efectos de este análisis, la cuenta de AWS que compartió la tabla es la cuenta del propietario y la cuenta con la que se compartió la tabla es la cuenta del destinatario. Cuando un trabajo de ETL en la cuenta del destinatario accede a los datos en la tabla de la cuenta del propietario, el evento CloudTrail de acceso a datos que se agrega a los registros de la cuenta del destinatario se copia en los registros de CloudTrail de la cuenta del propietario. Esto es para que las cuentas del propietario puedan realizar un seguimiento de los accesos a datos de las distintas cuentas de destinatarios. De forma predeterminada, los eventos de CloudTrail no incluyen un identificador principal inteligible (ARN principal). Un administrador en la cuenta del destinatario puede optar por incluir el ARN principal en los registros.

Para obtener más información, consulte Registro entre cuentas de CloudTrail en la Guía para desarrolladores de AWS Lake Formation.

Véase también

- the section called “Registro y monitoreo”

Propiedad de recursos entre cuentas y facturación

Cuando un usuario de una cuenta de AWS (cuenta A) crea un nuevo recurso como una base de datos en una cuenta diferente (cuenta B), dicho recurso pertenece a la cuenta B, la cuenta en la que se creó. Un administrador de la cuenta B obtiene automáticamente permisos completos para obtener acceso al nuevo recurso, como leer, escribir y conceder permisos de acceso a una tercera cuenta. El usuario de la cuenta A puede obtener acceso al recurso que acaba de crear solo si tienen los permisos adecuados concedidos por la cuenta B.

Los costos de almacenamiento y otros costos que guardan relación directa con el nuevo recurso se facturan a la cuenta B, el propietario de los recursos. El costo de las solicitudes de ese usuario que creó el recurso se factura a la cuenta del solicitante, la cuenta A.

Para obtener más información acerca de la facturación y los precios de AWS Glue, consulte Cómo funcionan los precios de AWS.

Limitaciones de acceso entre cuentas

El acceso entre cuentas de AWS Glue tiene las siguientes limitaciones:

- No se permite el acceso entre cuentas a AWS Glue si ha creado bases de datos y tablas con Amazon Athena o Amazon Redshift Spectrum antes de tener la compatibilidad de una región con AWS Glue y la cuenta del propietario de los recursos no ha migrado el catálogo de datos de Amazon Athena a AWS Glue. Puede encontrar el estado de migración actual utilizando GetCatalogImportStatus (get_catalog_import_status). Para obtener más información sobre cómo migrar un catálogo de Athena a AWS Glue, consulte Actualización a AWS Glue Data Catalog paso a paso en la Guía del usuario de Amazon Athena.
- El acceso entre cuentas solo se soporta para los recursos de Data Catalog, como bases de datos, tablas, funciones definidas por el usuario y conexiones.
- El acceso entre cuentas a Data Catalog de Athena exige que registre el catálogo como un recurso `DataCatalog` de Athena. Para obtener instrucciones, consulte Registro de un AWS Glue Data Catalog desde otra cuenta en la Guía del usuario de Amazon Athena.

Solución de problemas de identidad y acceso a AWS Glue

Usa la siguiente información para ayudarte a diagnosticar y solucionar problemas comunes que pueden surgir al trabajar con AWS Glue e IAM.

Temas

- No estoy autorizado a realizar ninguna acción en AWS Glue
- No estoy autorizado a realizar tareas como: PassRole
- Quiero permitir que personas ajenas a mí accedan Cuenta de AWS a mis recursos de AWS Glue

No estoy autorizado a realizar ninguna acción en AWS Glue

Si recibe un error que indica que no tiene autorización para realizar una acción, las políticas se deben actualizar para permitirle realizar la acción.

En el siguiente ejemplo, el error se produce cuando el usuario de IAM `mateojackson` intenta utilizar la consola para consultar los detalles acerca de un recurso ficticio *`my-example-widget`*, pero no tiene los permisos ficticios `glue:`*`GetWidget`*.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 glue:GetWidget on resource: my-example-widget
```

En este caso, la política del usuario `mateojackson` debe actualizarse para permitir el acceso al recurso *`my-example-widget`* mediante la acción `glue:`*`GetWidget`*.

Si necesitas ayuda, ponte en contacto con tu AWS administrador. El administrador es la persona que le proporcionó las credenciales de inicio de sesión.

No estoy autorizado a realizar tareas como: PassRole

Si recibes un error que indica que no estás autorizado a realizar la `iam:PassRole` acción, debes actualizar tus políticas para que puedas transferir una función a AWS Glue.

Algunas te Servicios de AWS permiten transferir una función existente a ese servicio en lugar de crear una nueva función de servicio o una función vinculada al servicio. Para ello, debe tener permisos para transferir el rol al servicio.

En el siguiente ejemplo, el error se produce cuando un usuario de IAM denominado `marymajor` intenta utilizar la consola para realizar una acción en AWS Glue. Sin embargo, la acción requiere

que el servicio cuente con permisos que otorguen un rol de servicio. Mary no tiene permisos para transferir el rol al servicio.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

En este caso, las políticas de Mary se deben actualizar para permitirle realizar la acción `iam:PassRole`.

Si necesita ayuda, póngase en contacto con su administrador. AWS El administrador es la persona que le proporcionó las credenciales de inicio de sesión.

Quiero permitir que personas ajenas a mí accedan Cuenta de AWS a mis recursos de AWS Glue

Puede crear un rol que los usuarios de otras cuentas o las personas externas a la organización puedan utilizar para acceder a sus recursos. Puede especificar una persona de confianza para que asuma el rol. En el caso de los servicios que admitan las políticas basadas en recursos o las listas de control de acceso (ACL), puede utilizar dichas políticas para conceder a las personas acceso a sus recursos.

Para más información, consulte lo siguiente:

- Para saber si AWS Glue admite estas funciones, consulteCómo funciona AWS Glue con IAM.
- Para obtener información sobre cómo proporcionar acceso a los recursos de su Cuentas de AWS propiedad, consulte Proporcionar acceso a un usuario de IAM en otro usuario de su propiedad Cuenta de AWS en la Guía del usuario de IAM.
- Para obtener información sobre cómo proporcionar acceso a tus recursos a terceros Cuentas de AWS, consulta Cómo proporcionar acceso a recursos que Cuentas de AWS son propiedad de terceros en la Guía del usuario de IAM.
- Para obtener información sobre cómo proporcionar acceso mediante una federación de identidades, consulte Proporcionar acceso a usuarios autenticados externamente (identidad federada) en la Guía del usuario de IAM.
- Para obtener información acerca del uso de roles y políticas basadas en recursos para el acceso entre cuentas, consulte Acceso a recursos entre cuentas en IAM en la Guía del usuario de IAM.

Registro y monitorización en AWS Glue

Puede automatizar la ejecución de sus trabajos de ETL (extracción, transformación y carga). AWS Glue proporciona métricas de rastreadores y trabajos que se pueden monitorear. Después de configurar AWS Glue Data Catalog con los metadatos necesarios, AWS Glue proporciona estadísticas sobre el estado de su entorno. Puede automatizar la invocación de rastreadores y trabajos con un programa basado en tiempo que esté basado en cron. También puede activar trabajos cuando se desencadena un disparador basado en eventos.

AWS Glue se integra con AWS CloudTrail, un servicio que proporciona un registro de las acciones realizadas por un usuario, un rol o un servicio de AWS en AWS Glue. Si crea un registro de seguimiento, puede habilitar la entrega continua de los eventos de CloudTrail a un bucket de Amazon Simple Storage Service (Amazon S3), Amazon CloudWatch Logs y Amazon CloudWatch Events. Cada entrada de registro o evento contiene información sobre quién generó la solicitud.

Utilice Amazon CloudWatch Events para automatizar los servicios de AWS y responder automáticamente a eventos del sistema, como problemas de disponibilidad de aplicaciones o cambios de recursos. Los eventos de los servicios de AWS se envían a CloudWatch Events casi en tiempo real. Puede crear reglas sencillas para indicar qué eventos resultan de interés, así como qué acciones automatizadas se van a realizar cuando un evento cumple una de las reglas.

Véase también

- Automatización de AWS Glue con CloudWatch Events
- Registro entre cuentas de CloudTrail

Un aspecto importante de la seguridad en la nube es el registro. Se debe configurar el registro de forma que no capture secretos ni material confidencial mientras captura la información necesaria para depurar y proteger la infraestructura en la nube. Asegúrese de familiarizarse con los elementos que se vayan a registrar.

Validación de conformidad para AWS Glue

Para saber si uno Servicio de AWS está dentro del ámbito de aplicación de programas de cumplimiento específicos, consulte Servicios de AWS Alcance por programa de cumplimiento

Servicios de AWS de cumplimiento y elija el programa de cumplimiento que le interese. Para obtener información general, consulte Programas de AWS cumplimiento > Programas AWS .

Puede descargar informes de auditoría de terceros utilizando AWS Artifact. Para obtener más información, consulte Descarga de informes en AWS Artifact .

Su responsabilidad de cumplimiento al Servicios de AWS utilizarlos viene determinada por la confidencialidad de sus datos, los objetivos de cumplimiento de su empresa y las leyes y reglamentos aplicables. AWS proporciona los siguientes recursos para ayudar con el cumplimiento:

- Guías de inicio rápido sobre seguridad y cumplimiento: estas guías de implementación analizan las consideraciones arquitectónicas y proporcionan los pasos para implementar entornos básicos centrados en AWS la seguridad y el cumplimiento.
- Diseño de arquitectura para garantizar la seguridad y el cumplimiento de la HIPAA en Amazon Web Services: en este documento técnico se describe cómo pueden utilizar AWS las empresas para crear aplicaciones aptas para la HIPAA.

  > **Note**
  >
  > No Servicios de AWS todas cumplen con los requisitos de la HIPAA. Para más información, consulte la Referencia de servicios compatibles con HIPAA.

- AWS Recursos de de cumplimiento: esta colección de libros de trabajo y guías puede aplicarse a su industria y ubicación.
- AWS Guías de cumplimiento para clientes: comprenda el modelo de responsabilidad compartida desde el punto de vista del cumplimiento. Las guías resumen las mejores prácticas para garantizar la seguridad Servicios de AWS y orientan los controles de seguridad en varios marcos (incluidos el Instituto Nacional de Estándares y Tecnología (NIST), el Consejo de Normas de Seguridad del Sector de Tarjetas de Pago (PCI) y la Organización Internacional de Normalización (ISO)).
- Evaluación de los recursos con reglas en la guía para AWS Config desarrolladores: el AWS Config servicio evalúa en qué medida las configuraciones de los recursos cumplen con las prácticas internas, las directrices del sector y las normas.
- AWS Security Hub— Esto Servicio de AWS proporciona una visión completa del estado de su seguridad interior AWS. Security Hub utiliza controles de seguridad para evaluar sus recursos de AWS y comprobar su cumplimiento con los estándares y las prácticas recomendadas del sector de la seguridad. Para obtener una lista de los servicios y controles compatibles, consulte la Referencia de controles de Security Hub.

- Amazon GuardDuty: Servicio de AWS detecta posibles amenazas para sus cargas de trabajo Cuentas de AWS, contenedores y datos mediante la supervisión de su entorno para detectar actividades sospechosas y maliciosas. GuardDuty puede ayudarlo a cumplir con varios requisitos de conformidad, como el PCI DSS, al cumplir con los requisitos de detección de intrusiones exigidos por ciertos marcos de cumplimiento.
- AWS Audit Manager— Esto le Servicio de AWS ayuda a auditar continuamente su AWS uso para simplificar la gestión del riesgo y el cumplimiento de las normativas y los estándares del sector.

Resiliencia en AWS Glue

La infraestructura AWS global se basa en AWS regiones y zonas de disponibilidad. AWS Las regiones proporcionan varias zonas de disponibilidad aisladas y separadas físicamente, que están conectadas mediante redes de baja latencia, alto rendimiento y alta redundancia. Con las zonas de disponibilidad, puede diseñar y utilizar aplicaciones y bases de datos que realizan una conmutación por error automática entre zonas de disponibilidad sin interrupciones. Las zonas de disponibilidad tienen una mayor disponibilidad, tolerancia a errores y escalabilidad que las infraestructuras tradicionales de centros de datos únicos o múltiples.

Para obtener más información sobre AWS las regiones y las zonas de disponibilidad, consulte Infraestructura global.AWS

Para obtener más información sobre la resiliencia de los AWS Glue trabajos, consulte Error: comportamiento de conmutación por error entre VPC en. AWS Glue

Seguridad de la infraestructura en AWS Glue

Al tratarse de un servicio administrado, AWS Glue está protegido por los procedimientos de seguridad de red globales de AWS que se describen en el documento técnico Amazon Web Services: Información general sobre los procesos de seguridad.

Puede utilizar llamadas a la API publicadas en AWS para obtener acceso a AWS Glue a través de la red. Los clientes deben ser compatibles con la seguridad de la capa de transporte (TLS) 1.0 o una versión posterior. Recomendamos TLS 1.2 o una versión posterior. Los clientes también deben ser compatibles con conjuntos de cifrado con confidencialidad directa total (PFS) tales como Ephemeral Diffie-Hellman (DHE) o Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). La mayoría de los sistemas modernos como Java 7 y posteriores son compatibles con estos modos.

Además, las solicitudes deben estar firmadas mediante un ID de clave de acceso y una clave de acceso secreta que esté asociada a una entidad principal de IAM. También puede utilizar AWS Security Token Service (AWS STS) para generar credenciales de seguridad temporales para firmar solicitudes.

Temas

- AWS Glue y puntos de conexión de VPC de interfaz (AWS PrivateLink)
- VPC de Amazon compartidas

AWS Glue y puntos de conexión de VPC de interfaz (AWS PrivateLink)

Puede establecer una conexión privada entre la VPC y AWS Glue mediante la creación de un punto de conexión de VPC de interfaz. Los puntos de conexión de interfaz cuentan con tecnología de AWS PrivateLink que le permite acceder de forma privada a las API de AWS Glue sin una puerta de enlace de Internet, un dispositivo NAT, una conexión de VPN o una conexión de AWS Direct Connect. Las instancias de su VPC no necesitan direcciones IP públicas para comunicar con las API de AWS Glue. El tráfico entre la VPC y AWS Glue no sale de la red de Amazon.

Cada punto de conexión de interfaz está representado por una o más interfaces de red elásticas en las subredes.

Para obtener más información, consulte puntos de conexión de VPC de interfaz (AWS PrivateLink) en la Guía del usuario de Amazon VPC.

Consideraciones para los puntos de conexión de VPC de AWS Glue

Antes de configurar un punto de conexión de VPC de tipo interfaz para AWS Glue, asegúrese de revisar las Propiedades y limitaciones de los puntos de conexión de interfaz en la Guía del usuario de Amazon VPC.

AWS Glue admite realizar llamadas a todas sus acciones de la API desde su VPC.

Creación de un punto de conexión de VPC de interfaz para AWS Glue

Puede crear un punto de conexión de VPC para el servicio de AWS Glue mediante la consola de Amazon VPC o AWS Command Line Interface (AWS CLI). Para más información, consulte Creación de un punto de conexión de interfaz en la Guía del usuario de Amazon VPC.

Cree un punto de conexión de VPC para AWS Glue, mediante el siguiente nombre de servicio:

- com.amazonaws.*region*.glue

Si habilita DNS privado para el punto de conexión, puede realizar solicitudes a la API para AWS Glue usando su nombre de DNS predeterminado para la región, por ejemplo `glue.us-east-1.amazonaws.com`.

Para más información, consulte Acceso a un servicio a través de un punto de conexión de interfaz en la Guía del usuario de Amazon VPC.

Creación de una política de puntos de conexión de VPC para AWS Glue

Puede asociar una política de punto de conexión con su punto de conexión de VPC que controla el acceso a AWS Glue. La política especifica la siguiente información:

- La entidad principal que puede realizar acciones.
- Las acciones que se pueden realizar.
- Los recursos en los que se pueden llevar a cabo las acciones.

Para más información, consulte Control del acceso a los servicios con puntos de enlace de la VPC en la Guía del usuario de Amazon VPC.

Ejemplo: política de punto de conexión de VPC para que AWS Glue permita la creación y actualización de trabajos

A continuación, se muestra un ejemplo de una política de puntos de conexión de AWS Glue. Cuando se asocia con un punto de conexión, esta política concede acceso a las acciones de AWS Glue mostradas para todas las entidades principales en todos los recursos.

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
             "glue:CreateJob",
             "glue:UpdateJob",
             "iam:PassRole"
         ],
         "Resource":"*"
      }
```

```
    ]
}
```

Ejemplo: política de punto de conexión de VPC para permitir acceso de solo lectura a Data Catalog

A continuación, se muestra un ejemplo de una política de puntos de conexión de AWS Glue. Cuando se asocia con un punto de conexión, esta política concede acceso a las acciones de AWS Glue mostradas para todas las entidades principales en todos los recursos.

```
{
    "Statement": [
        {
            "Principal": "*",
            "Effect": "Allow",
            "Action": [
                "glue:GetDatabase",
                "glue:GetDatabases",
                "glue:GetTable",
                "glue:GetTables",
                "glue:GetTableVersion",
                "glue:GetTableVersions",
                "glue:GetPartition",
                "glue:GetPartitions",
                "glue:BatchGetPartition",
                "glue:SearchTables"
            ],
            "Resource": "*"
        }
    ]
}
```

VPC de Amazon compartidas

AWS Glue soporta las nubes privadas virtuales (VPC) compartidas en Amazon Virtual Private Cloud. El uso compartido de Amazon VPC permite que varias cuentas de AWS creen sus recursos de aplicaciones, como instancias de Amazon EC2 y bases de datos de Amazon Relational Database Service (Amazon RDS), en VPC de Amazon compartidas y administradas de forma centralizada. En este modelo, la cuenta propietaria de la VPC (el propietario) comparte una o varias subredes con otras cuentas (los participantes) que pertenecen a la misma organización de AWS Organizations. Después de compartir una subred, los participantes pueden ver, crear, modificar y eliminar los recursos de su aplicación en las subredes compartidas con ellos.

En AWS Glue, para crear una conexión con una subred compartida, debe crear un grupo de seguridad dentro de su cuenta y asociar el grupo de seguridad a la subred compartida.

Para obtener más información, consulte estos temas:

- Trabajar con VPC compartidas en la Guía del usuario de Amazon VPC
- ¿Qué es AWS Organizations? en la Guía del usuario de AWS Organizations

Solución de problemas de AWS Glue

Temas

- Recopilación de información sobre la solución de problemas en AWS Glue
- Solución de errores en AWS Glue Spark
- Solución de problemas AWS Glue de errores de Ray en los registros
- Excepciones de machine learning en AWS Glue
- Cuotas de AWS Glue

Recopilación de información sobre la solución de problemas en AWS Glue

Si detecta errores o comportamientos no esperados en AWS Glue y debe ponerse en contacto con AWS Support, en primer lugar, deberá recopilar la información sobre los nombres, los ID y los registros asociados con el error. Con esta información, AWS Support podrán ayudarlo a solucionar los problemas que experimenta.

Junto con su ID de cuenta, recopile la siguiente información para cada uno de estos tipos de errores:

Cuando se produzca un error en un rastreador, recopile la siguiente información:

- Nombre del rastreador

  Los registros de las ejecuciones del rastreador se encuentran en CloudWatch Logs en `/aws-glue/crawlers`.

Cuando se produzca un error en una conexión de prueba, recopile la siguiente información:

- Connection name
- ID de la conexión
- Cadena de la conexión de JDBC con formato `jdbc:protocol://host:port/database-name`.

  Los registros de las conexiones de prueba se encuentra en CloudWatch Logs en `/aws-glue/testconnection`.

Cuando se produzca un error en un flujo de trabajo, recopile la siguiente información:

- Nombre del trabajo

- ID de ejecución del flujo de trabajo con el formato `jr_xxxxx`.

  Los registros de las ejecuciones de trabajo se encuentran en CloudWatch Logs en `/aws-glue/jobs`.

Solución de errores en AWS Glue Spark

Si encuentra errores en AWS Glue, utilice las siguientes soluciones para encontrar el origen de los problemas y solucionarlos.

> **Note**
>
> El AWS Glue GitHub repositorio contiene instrucciones adicionales para la solución de problemas en las AWS Glue Preguntas frecuentes.

Temas

- Error: Resource unavailable (Recurso no disponible)
- Error: Could Not Find S3 Endpoint or NAT Gateway for subnetId in VPC (No se encontró el punto de conexión S3 o puerta de enlace NAT para subnetId en la VPC)
- Error: Inbound Rule in Security Group Required (La regla de entrada del grupo de seguridad es obligatoria)
- Error: Outbound Rule in Security Group Required (La regla de salida del grupo de seguridad es obligatoria)
- Error: la ejecución del trabajo ha fallado porque el rol transferido debe tener permisos de asuma el rol para el AWS Glue servicio
- Error: la DescribeVpcEndpoints acción no está autorizada. No se puede validar el ID de VPC vpc-id
- Error: la DescribeRouteTables acción no está autorizada. No se pudo validar el ID de subred: ID de subred en el ID de VPC: vpc-id
- Error: no se pudo llamar a ec2: DescribeSubnets
- Error: no se pudo llamar a ec2: DescribeSecurityGroups
- Error: Could Not Find Subnet for AZ (No se pudo encontrar la subred de AZ)
- Error: Job run exception when writing to a JDBC target (Excepción de ejecución de flujo de trabajo al escribir en un destino JDBC)

- Error: Amazon S3: The operation is not valid for the object's storage class
- Error: Amazon S3 timeout (Tiempo de espera de Amazon S3)
- Error: Amazon S3 access denied (Acceso denegado a Amazon S3)
- Error: Amazon S3 access key ID does not exist (El ID de clave de acceso de Amazon S3 no existe)
- Se produce un error en la ejecución de trabajo al obtener acceso a Amazon S3 con un URI de s3a://
- Error: Amazon S3 service token expired (Token de servicio de Amazon S3 caducado)
- Error: No private DNS for network interface found (No se encontró un DNS privado para la interfaz de red)
- Error: Development endpoint provisioning failed (Error en el aprovisionamiento del punto de conexión de desarrollo)
- Error: Notebook Server CREATE_FAILED (CREATE_FAILED el servidor del bloc de notas)
- Error: Local notebook fails to start (El bloc de notas local no se inicia)
- Error: Running crawler failed (Error de ejecución del rastreador)
- Error: Partitions were not updated (Las particiones no se han actualizado)
- Error: no se pudo actualizar el marcador de trabajo debido a que la versión no coincide
- Error: A job is reprocessing data when job bookmarks are enabled (Un trabajo está reprocesando datos cuando los marcadores de trabajo están habilitados)
- Error: comportamiento de conmutación por error entre VPC en AWS Glue
- Solucionar los errores del rastreador cuando éste utiliza las credenciales de Lake Formation

Error: Resource unavailable (Recurso no disponible)

Si AWS Glue devuelve el mensaje de que un recurso no está disponible, puedes ver los mensajes de error o los registros para obtener más información sobre el problema. Las siguientes tareas describen métodos generales para solucionar problemas.

- Por cada conexión o punto de enlace de desarrollo que utilice, compruebe que el clúster no se haya quedado sin interfaces de red elásticas.

Error: Could Not Find S3 Endpoint or NAT Gateway for subnetId in VPC (No se encontró el punto de conexión S3 o puerta de enlace NAT para subnetId en la VPC)

Compruebe el ID de la subred y el ID de la VPC indicados en el mensaje como ayuda para diagnosticar el problema.

- Compruebe que tenga un punto de conexión de VPC de Amazon S3 configurado, elemento obligatorio con AWS Glue. Además, compruebe la gateway de NAT si forma parte de su configuración. Para obtener más información, consulte Puntos de enlace de Amazon VPC para Amazon S3.

Error: Inbound Rule in Security Group Required (La regla de entrada del grupo de seguridad es obligatoria)

Al menos un grupo de seguridad debe abrir todos los puertos de entrada. Para limitar el tráfico, el grupo de seguridad de origen de su regla de entrada puede restringirse al mismo grupo de seguridad.

- En todas las conexiones que utilice, compruebe que el grupo de seguridad tenga una regla de entrada con autorreferencia. Para obtener más información, consulte Configuración del acceso de red a los almacenes de datos.
- Cuando utilice un punto de enlace de desarrollo, compruebe que su grupo de seguridad tenga una regla de entrada con autorreferencia. Para obtener más información, consulte Configuración del acceso de red a los almacenes de datos.

Error: Outbound Rule in Security Group Required (La regla de salida del grupo de seguridad es obligatoria)

Al menos un grupo de seguridad debe abrir todos los puertos de salida. Para limitar el tráfico, el grupo de seguridad de origen de su regla de salida puede restringirse al mismo grupo de seguridad.

- En todas las conexiones que utilice, compruebe que el grupo de seguridad tenga una regla de salida que se haga referencia a sí misma. Para obtener más información, consulte Configuración del acceso de red a los almacenes de datos.

- Cuando utilice un punto de enlace de desarrollo, compruebe que su grupo de seguridad tenga una regla de salida que se haga referencia a sí misma. Para obtener más información, consulte Configuración del acceso de red a los almacenes de datos.

Error: la ejecución del trabajo ha fallado porque el rol transferido debe tener permisos de asuma el rol para el AWS Glue servicio

El usuario que define un flujo de trabajo debe tener permiso para `iam:PassRole` para AWS Glue.

- Cuando un usuario crea un AWS Glue trabajo, confirme que el rol del usuario contenga una política que contenga `iam:PassRole` for AWS Glue. Para obtener más información, consulte Paso 3: Adjuntar una política a los usuarios o los grupos que accedan a AWS Glue.

Error: la DescribeVpcEndpoints acción no está autorizada. No se puede validar el ID de VPC vpc-id

- Comprueba la política a la que se ha transferido el permiso. AWS Glue `ec2:DescribeVpcEndpoints`

Error: la DescribeRouteTables acción no está autorizada. No se pudo validar el ID de subred: ID de subred en el ID de VPC: vpc-id

- Comprueba la política a AWS Glue la `ec2:DescribeRouteTables` que se ha transferido el permiso.

Error: no se pudo llamar a ec2: DescribeSubnets

- Compruebe la política a la que se ha AWS Glue enviado el `ec2:DescribeSubnets` permiso.

Error: no se pudo llamar a ec2: DescribeSecurityGroups

- Compruebe la política a la que se ha AWS Glue enviado el `ec2:DescribeSecurityGroups` permiso.

Error: Could Not Find Subnet for AZ (No se pudo encontrar la subred de AZ)

- Es posible que la zona de disponibilidad no esté disponible para AWS Glue. Cree y utilice una nueva subred en una zona de disponibilidad que no sea la que se ha especificado en el mensaje.

Error: Job run exception when writing to a JDBC target (Excepción de ejecución de flujo de trabajo al escribir en un destino JDBC)

Cuando ejecuta un flujo de trabajo que escribe en un destino JDBC, es posible que el flujo de trabajo presente errores en los siguientes casos:

- Si su flujo de trabajo escribe en una tabla de Microsoft SQL Server y la tabla tiene definidas columnas de tipo `Boolean`, la tabla tiene que definirse previamente en la base de datos de SQL Server. Cuando defina el trabajo en la AWS Glue consola mediante un destino de SQL Server con la opción Crear tablas en el destino de datos, no asigne ninguna columna de origen a una columna de destino con un tipo de datos`Boolean`. Es posible que se produzca un error cuando el flujo de trabajo se ejecute.

  Para evitar el error, ejecute la siguiente operación:

  - Elija una tabla existente con la columna Boolean (booleana).
  - Edite la transformación `ApplyMapping` y mapee la columna Boolean del origen con un número o una cadena del destino.
  - Edite la transformación `ApplyMapping` para eliminar la columna Boolean del origen.
- Si su flujo de trabajo escribe en una tabla de Oracle, es posible que deba ajustar la longitud de los nombres de los objetos de Oracle. En algunas versiones de Oracle, la longitud de identificador máxima está limitada a 30 o 128 bytes. Este límite afecta a los nombres de tablas y de columnas de los almacenes de datos de destino de Oracle.

  Para evitar el error, ejecute la siguiente operación:

  - Dé un nombre a las tablas de destino de Oracle que esté dentro de los límites de su versión.
  - Los nombres de columna predeterminados se generan a partir de los nombres de campos de los datos. Para gestionar el error cuando los nombres de las columnas superan el límite establecido, utilice las transformaciones `ApplyMapping` o `RenameField` para cambiar el nombre de la columna para que esté dentro del límite aceptado.

Error: Amazon S3: The operation is not valid for the object's storage class

Si AWS Glue devuelve este error, es posible que su AWS Glue trabajo haya estado leyendo datos de tablas que tienen particiones en todos los niveles de clases de almacenamiento de Amazon S3.

- Al utilizar las exclusiones de clases de almacenamiento, puede asegurarse de que sus AWS Glue trabajos funcionarán en tablas que tengan particiones en estos niveles de clases de almacenamiento. Sin exclusiones, los trabajos que leen datos de estos niveles fallan con el siguiente error: `AmazonS3Exception: The operation is not valid for the object's storage class`.

  Para obtener más información, consulte Exclusión de clases de almacenamiento de Amazon S3.

Error: Amazon S3 timeout (Tiempo de espera de Amazon S3)

Si AWS Glue devuelve un error en el que se ha agotado el tiempo de espera de la conexión, puede deberse a que está intentando acceder a un bucket de Amazon S3 en otra AWS región.

- Un punto de enlace de VPC de Amazon S3 solo puede enrutar el tráfico a buckets dentro de una región. AWS Si necesita conectarse a buckets de otras regiones, puede recurrir a una gateway de NAT para evitar este problema. Para obtener más información, consulte Puerta de enlace NAT.

Error: Amazon S3 access denied (Acceso denegado a Amazon S3)

Si AWS Glue devuelve un error de acceso denegado a un bucket u objeto de Amazon S3, puede deberse a que la función de IAM proporcionada no tiene una política que autorice su almacén de datos.

- Un flujo de trabajo de ETL debe tener acceso al almacén de datos de Amazon S3 utilizado como origen o destino. Un rastreador debe tener acceso al almacén de datos de Amazon S3 que rastrea. Para obtener más información, consulte Paso 2: creación de un rol de IAM para AWS Glue.

Error: Amazon S3 access key ID does not exist (El ID de clave de acceso de Amazon S3 no existe)

Si AWS Glue devuelve un error con el identificador de la clave de acceso y no existe al ejecutar un trabajo, puede deberse a uno de los siguientes motivos:

- Un trabajo de ETL utiliza un rol de IAM para obtener acceso a los almacenes de datos. Confirme que el rol de IAM de su trabajo no se haya eliminado antes de iniciar el trabajo.
- Un rol de IAM contiene permisos para obtener acceso a sus almacenes de datos. Confirme que todas las políticas de Amazon S3 asociadas que contengan `s3:ListBucket` sean correctas.

Se produce un error en la ejecución de trabajo al obtener acceso a Amazon S3 con un URI de **`s3a://`**

Si un trabajo devuelve un error similar a Failed to parse XML document with handler class (No se pudo analizar documento XML con la clase de controlador), podría deberse a un error al intentar enumerar centenares de archivos mediante un URI `s3a://`. Obtenga acceso a su almacén de datos mediante un URI `s3://` en su lugar. En el siguiente seguimiento de excepciones se destacan los errores que se buscan:

```
1. com.amazonaws.SdkClientException: Failed to parse XML document with handler class
 com.amazonaws.services.s3.model.transform.XmlResponsesSaxParser$ListBucketHandler
2. at
 com.amazonaws.services.s3.model.transform.XmlResponsesSaxParser.parseXmlInputStream(XmlRespons
3. at
 com.amazonaws.services.s3.model.transform.XmlResponsesSaxParser.parseListBucketObjectsResponse
4. at com.amazonaws.services.s3.model.transform.Unmarshallers
$ListObjectsUnmarshaller.unmarshall(Unmarshallers.java:70)
5. at com.amazonaws.services.s3.model.transform.Unmarshallers
$ListObjectsUnmarshaller.unmarshall(Unmarshallers.java:59)
6. at
 com.amazonaws.services.s3.internal.S3XmlResponseHandler.handle(S3XmlResponseHandler.java:62)
7. at
 com.amazonaws.services.s3.internal.S3XmlResponseHandler.handle(S3XmlResponseHandler.java:31)
8. at
 com.amazonaws.http.response.AwsResponseHandlerAdapter.handle(AwsResponseHandlerAdapter.java:70
9. at com.amazonaws.http.AmazonHttpClient
$RequestExecutor.handleResponse(AmazonHttpClient.java:1554)
10. at com.amazonaws.http.AmazonHttpClient
$RequestExecutor.executeOneRequest(AmazonHttpClient.java:1272)
11. at com.amazonaws.http.AmazonHttpClient
$RequestExecutor.executeHelper(AmazonHttpClient.java:1056)
12. at com.amazonaws.http.AmazonHttpClient
$RequestExecutor.doExecute(AmazonHttpClient.java:743)
```

```
13. at com.amazonaws.http.AmazonHttpClient
$RequestExecutor.executeWithTimer(AmazonHttpClient.java:717)
14. at com.amazonaws.http.AmazonHttpClient
$RequestExecutor.execute(AmazonHttpClient.java:699)
15. at com.amazonaws.http.AmazonHttpClient$RequestExecutor.access
$500(AmazonHttpClient.java:667)
16. at com.amazonaws.http.AmazonHttpClient
$RequestExecutionBuilderImpl.execute(AmazonHttpClient.java:649)
17. at com.amazonaws.http.AmazonHttpClient.execute(AmazonHttpClient.java:513)
18. at com.amazonaws.services.s3.AmazonS3Client.invoke(AmazonS3Client.java:4325)
19. at com.amazonaws.services.s3.AmazonS3Client.invoke(AmazonS3Client.java:4272)
20. at com.amazonaws.services.s3.AmazonS3Client.invoke(AmazonS3Client.java:4266)
21. at com.amazonaws.services.s3.AmazonS3Client.listObjects(AmazonS3Client.java:834)
22. at org.apache.hadoop.fs.s3a.S3AFileSystem.getFileStatus(S3AFileSystem.java:971)
23. at
 org.apache.hadoop.fs.s3a.S3AFileSystem.deleteUnnecessaryFakeDirectories(S3AFileSystem.java:115
24. at org.apache.hadoop.fs.s3a.S3AFileSystem.finishedWrite(S3AFileSystem.java:1144)
25. at org.apache.hadoop.fs.s3a.S3AOutputStream.close(S3AOutputStream.java:142)
26. at org.apache.hadoop.fs.FSDataOutputStream
$PositionCache.close(FSDataOutputStream.java:74)
27. at org.apache.hadoop.fs.FSDataOutputStream.close(FSDataOutputStream.java:108)
28. at org.apache.parquet.hadoop.ParquetFileWriter.end(ParquetFileWriter.java:467)
29. at
 org.apache.parquet.hadoop.InternalParquetRecordWriter.close(InternalParquetRecordWriter.java:1
30. at
 org.apache.parquet.hadoop.ParquetRecordWriter.close(ParquetRecordWriter.java:112)
31. at
 org.apache.spark.sql.execution.datasources.parquet.ParquetOutputWriter.close(ParquetOutputWrit
32. at org.apache.spark.sql.execution.datasources.FileFormatWriter
$SingleDirectoryWriteTask.releaseResources(FileFormatWriter.scala:252)
33. at org.apache.spark.sql.execution.datasources.FileFormatWriter$$anonfun
$org$apache$spark$sql$execution$datasources$FileFormatWriter$$executeTask
$3.apply(FileFormatWriter.scala:191)
34. at org.apache.spark.sql.execution.datasources.FileFormatWriter$$anonfun
$org$apache$spark$sql$execution$datasources$FileFormatWriter$$executeTask
$3.apply(FileFormatWriter.scala:188)
35. at org.apache.spark.util.Utils
$.tryWithSafeFinallyAndFailureCallbacks(Utils.scala:1341)
36. at org.apache.spark.sql.execution.datasources.FileFormatWriter$.org$apache$spark
$sql$execution$datasources$FileFormatWriter$$executeTask(FileFormatWriter.scala:193)
37. at org.apache.spark.sql.execution.datasources.FileFormatWriter$$anonfun$write$1$
$anonfun$3.apply(FileFormatWriter.scala:129)
38. at org.apache.spark.sql.execution.datasources.FileFormatWriter$$anonfun$write$1$
$anonfun$3.apply(FileFormatWriter.scala:128)
```

```
39. at org.apache.spark.scheduler.ResultTask.runTask(ResultTask.scala:87)
40. at org.apache.spark.scheduler.Task.run(Task.scala:99)
41. at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:282)
42. at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
43. at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
44. at java.lang.Thread.run(Thread.java:748)
```

Error: Amazon S3 service token expired (Token de servicio de Amazon S3 caducado)

Al transferir datos hacia y desde Amazon Redshift, se utilizan las credenciales temporales de Amazon S3, que caducan después de 1 hora. Si tiene un trabajo de larga duración, se podría producir un error. Para obtener información acerca de cómo configurar sus trabajos de larga duración para mover datos hacia y desde Amazon Redshift, consulte aws-glue-programming-etl-connect-redshift-home.

Error: No private DNS for network interface found (No se encontró un DNS privado para la interfaz de red)

Si un flujo de trabajo genera un error o un punto de enlace de desarrollo no aprovisiona correctamente, podría ser debido a un problema en la configuración de la red.

- Si utiliza el DNS suministrado por Amazon, el valor de `enableDnsHostnames` debe establecerse en true. Para obtener más información, consulte DNS.

Error: Development endpoint provisioning failed (Error en el aprovisionamiento del punto de conexión de desarrollo)

Si AWS Glue no se aprovisiona correctamente un terminal de desarrollo, puede deberse a un problema en la configuración de la red.

- Cuando defina un punto de enlace de desarrollo, la VPC, la subred y los grupos de seguridad se validan para confirmar que cumplan determinados requisitos.
- Si ha proporcionado la clave pública SSH opcional, compruebe que esta sea una clave pública SSH válida.

- Consulte en la consola de la VPC que su VPC utilice la opción DHCP option set (Conjunto de opciones de DHCP) válida. Para obtener más información, consulte Conjuntos de opciones de DHCP.
- Si el clúster permanece en estado PROVISIONING (APROVISIONAMIENTO), póngase en contacto con AWS Support.

Error: Notebook Server CREATE_FAILED (CREATE_FAILED el servidor del bloc de notas)

Si AWS Glue no se crea el servidor portátil para un terminal de desarrollo, puede deberse a uno de los siguientes problemas:

- AWS Glue transfiere una función de IAM a Amazon EC2 cuando configura el servidor del portátil. El rol de IAM debe tener una relación de confianza con Amazon EC2.
- El rol de IAM debe tener un perfil de instancias con el mismo nombre. Cuando crea el rol para Amazon EC2 con la consola de IAM, se crea automáticamente el perfil de instancias con el mismo nombre. Compruebe si el registro contiene un error de nombre de perfil de instancias `iamInstanceProfile.name` no válido. Para obtener más información, consulte Utilización de perfiles de instancia.
- Compruebe que su rol tenga permiso para obtener acceso a buckets `aws-glue*` en la política que pasa para crear el servidor del bloc de notas.

Error: Local notebook fails to start (El bloc de notas local no se inicia)

Si el bloc de notas local no se inicia y notifica errores del tipo que un directorio o una carpeta no se pueden encontrar, podría ser debido a uno de los siguientes problemas:

- Si usa Microsoft Windows, asegúrese de que la variable de entorno `JAVA_HOME` apunte al directorio de Java correcto. Es posible actualizar Java sin actualizar esta variable; si la variable apunta a una carpeta que ya no existe, los cuadernos de Jupyter no se iniciarán.

Error: Running crawler failed (Error de ejecución del rastreador)

Si AWS Glue no se ejecuta correctamente un rastreador para catalogar los datos, puede deberse a uno de los siguientes motivos. En primer lugar compruebe si un error aparece en la lista de

rastreadores de la consola de AWS Glue . Compruebe si hay un icono de exclamación junto al nombre del rastreador y pase el ratón por encima del icono para ver si hay mensajes asociados.

- Compruebe los registros del rastreador ejecutado en CloudWatch Registros, en. `/aws-glue/crawlers`

Error: Partitions were not updated (Las particiones no se han actualizado)

En caso de que sus particiones no estuvieran actualizadas en el catálogo de datos cuando ejecutó un trabajo de ETL, estas instrucciones de registro de la `DataSink` clase incluida en los CloudWatch registros pueden resultarle útiles:

- "`Attempting to fast-forward updates to the Catalog - nameSpace:`": muestra qué base de datos, tabla e identificador de catálogo se intenta modificar en este trabajo. Si esta instrucción no está aquí, compruebe si `enableUpdateCatalog` está establecido en true y se ha pasado correctamente como un parámetro `getSink()` o en `additional_options`.
- "`Schema change policy behavior:`": muestra qué valor `updateBehavior` del esquema se ha pasado.
- "`Schemas qualify (schema compare):`": será true o false.
- "`Schemas qualify (case-insensitive compare):`": será true o false.
- Si ambos son falsos y el suyo no `updateBehavior` está establecido en él `UPDATE_IN_DATABASE`, el DynamicFrame esquema debe ser idéntico o contener un subconjunto de las columnas que aparecen en el esquema de la tabla del catálogo de datos.

Para obtener más información sobre actualización de particiones, consulte Cómo actualizar el esquema y añadir nuevas particiones al Catálogo de datos mediante trabajos de ETL de AWS Glue.

Error: no se pudo actualizar el marcador de trabajo debido a que la versión no coincide

Es posible que esté intentando parametizar AWS Glue trabajos para aplicar la misma transformación/lógica en diferentes conjuntos de datos en Amazon S3. Desea realizar un seguimiento de los archivos procesados en las ubicaciones proporcionadas. Cuando ejecuta el mismo trabajo en el mismo bucket de origen y escribe en el mismo destino o en uno diferente simultáneamente (concurrencia >1), el trabajo falla con este error:

```
py4j.protocol.Py4JJavaError: An error occurred while
 callingz:com.amazonaws.services.glue.util.Job.commit.:com.amazonaws.services.gluejobexecutor.m
 Continuation update failed due to version mismatch. Expected version 2 but found
 version 3
```

Solución: establezca la concurrencia en 1 o no ejecute el trabajo simultáneamente.

Actualmente, los AWS Glue marcadores no admiten la ejecución simultánea de tareas y las confirmaciones fallarán.

Error: A job is reprocessing data when job bookmarks are enabled (Un trabajo está reprocesando datos cuando los marcadores de trabajo están habilitados)

Puede haber casos en los que haya activado los marcadores de tareas, pero su AWS Glue tarea de ETL consista en reprocesar datos que ya se habían procesado en una ejecución anterior. Compruebe si hay estas causas comunes de este error:

Simultaneidad máxima

Si el número máximo establecido de ejecuciones concurrentes para el trabajo es mayor que el valor predeterminado de 1, esto puede interferir con los marcadores de trabajo. Esto puede suceder cuando los marcadores de trabajo comprueban la hora de la última modificación para verificar los objetos que deben volver a procesarse. Para obtener más información, consulte la discusión de simultaneidad máxima en Configurar las propiedades de los trabajos de Spark en AWS Glue.

Objeto de trabajo ausente

Asegúrese de que su script de ejecución de trabajo termina con la siguiente confirmación:

```
job.commit()
```

Al incluir este objeto, AWS Glue registra la marca de tiempo y la ruta de la ejecución del trabajo. Si vuelve a ejecutar el trabajo con la misma ruta, AWS Glue procesa solo los archivos nuevos. Si no incluye este objeto y los marcadores de trabajo están habilitados, el trabajo vuelve a procesar los archivos ya procesados junto con los nuevos archivos y crea redundancia en el almacén de datos de destino del trabajo.

Parámetro de contexto de transformación ausente

El contexto de transformación es un parámetro opcional en la clase `GlueContext`, pero los marcadores de trabajo no funcionan si no lo incluye. Para resolver este error, añada el parámetro de contexto de transformación al crear el DynamicFrame, como se muestra a continuación:

```
sample_dynF=create_dynamic_frame_from_catalog(database,
 table_name,transformation_ctx="sample_dynF")
```

Origen de entrada

Si utiliza una base de datos relacional (una conexión JDBC) para la fuente de entrada, los marcadores de trabajo funcionan solamente si las claves de principal de la tabla se encuentran en orden secuencial. Los marcadores de trabajo funcionan en nuevas filas, pero no en filas actualizadas. Esto se debe a que los marcadores de trabajo buscan las claves principales, que ya existen. Esto no se aplica si su origen de entrada es Amazon Simple Storage Service (Amazon S3).

Hora de la última modificación

Para orígenes de entrada de Amazon S3, los marcadores de trabajo comprueban la hora de última modificación de los objetos, en lugar de los nombres de archivo, para verificar qué objetos deben volver a procesarse. Si los datos de origen de entrada se han modificado desde la última ejecución de trabajo, los archivos se vuelven a procesar al ejecutar el trabajo de nuevo.

Error: comportamiento de conmutación por error entre VPC en AWS Glue

El siguiente proceso se utiliza para la conmutación por error de los trabajos de la AWS Glue versión 4.0 y versiones anteriores.

Resumen: se selecciona una AWS Glue conexión en el momento en que se envía la ejecución de un trabajo. Si la ejecución del trabajo presenta algunos problemas (falta de direcciones IP, conectividad al origen, problema de enrutamiento), la ejecución fallará. Si los reintentos están configurados, AWS Glue lo volverá a intentar con la misma conexión.

1. Para cada intento de ejecución, AWS Glue comprobará el estado de las conexiones en el orden indicado en la configuración del trabajo, hasta que encuentre una que pueda utilizar. En caso de que se produzca un error en la zona de disponibilidad (AZ), las conexiones procedentes de esa zona no superarán la comprobación y se omitirán.
2. AWS Glue valida la conexión con lo siguiente:
   - comprueba si el identificador y la subred de Amazon VPC son válidos.

- comprueba que existe una puerta de enlace NAT o un punto de conexión de VPC de Amazon.
- comprueba que la subred tiene más de 0 direcciones IP asignadas.
- comprueba que la AZ esté en buen estado.

AWS Glue no puede verificar la conectividad en el momento del envío de la ejecución del trabajo.

3. En el caso de los trabajos que utilizan Amazon VPC, todos los controladores y ejecutores se crearán en la misma zona de disponibilidad con la conexión seleccionada en el momento de enviar el trabajo.
4. Si los reintentos están configurados, AWS Glue lo volverá a intentar con la misma conexión. Esto se debe a que no podemos garantizar que los problemas con esta conexión se prolonguen durante mucho tiempo. Si una zona de disponibilidad falla, el trabajo existente que se ejecuta (según la etapa de ejecución del trabajo) en esa zona puede fallar. Al volver a intentarlo, se debe detectar un error en la zona de disponibilidad y elegir otra zona de disponibilidad para la nueva ejecución.

Solucionar los errores del rastreador cuando éste utiliza las credenciales de Lake Formation

Utilice la siguiente información para diagnosticar y solucionar varios problemas al configurar el rastreador mediante las credenciales de Lake Formation.

Error: la ubicación de S3: s3://examplepath no está registrada

Para que un rastreador se ejecute con las credenciales de Lake Formation, primero debe configurar los permisos de Lake Formation. Para resolver este error, registre la ubicación de Amazon S3 de destino con Lake Formation. Para obtener más información, consulte Registro de una ubicación de Amazon S3.

Error: el usuario o rol no está autorizado a realizar: lakeformation:GetDataAccess en el recurso

Añada el permiso `lakeformation:GetDataAccess` del rol de rastreador con la consola de IAM o AWS CLI. Con este permiso, Lake Formation concede la solicitud de credenciales temporales para acceder a los datos. Consulte la política siguiente:

```
{
```

```
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "lakeformation:GetDataAccess"
    ],
    "Resource": "*"
  }
}
```

Error: permisos insuficientes para la formación de lagos en (Nombre de la base de datos: exampleDatabase, Nombre de la tabla: exampleTable)

En la consola de Lake Formation (https://console.aws.amazon.com/lakeformation/), conceda permisos de acceso al rol de rastreador (`Create`, `Describe`, `Alter`) en la base de datos, que se especifica como la base de datos de salida. También puede conceder permisos a la tabla. Para obtener más información, consulte Concesión de permisos de base de datos mediante la consola de Lake Formation y el método de recurso con nombre.

Error: permisos de Lake Formation insuficientes en s3://examplepath

1. Rastreo entre cuentas
   a. Inicie sesión en la consola de Lake Formation (https://console.aws.amazon.com/lakeformation/) mediante la cuenta en la que está registrado el bucket de Amazon S3 (cuenta B). Conceda permisos de Data location (Ubicación de datos) a la cuenta en la que se vaya a ejecutar el rastreador. Esto permitirá al rastreador leer los datos de la ubicación de Amazon S3 de destino.
   b. En la cuenta donde esté creado el rastreador (cuenta A), conceda a Data location (ubicación de los datos) permisos en la ubicación de Amazon S3 de destino para el rol de IAM que se usa para la ejecución del rastreador, de modo que el rastreador pueda leer los datos del destino en Lake Formation. Para más información, consulte Concesión de permisos de ubicación de datos (misma cuenta).
2. Rastreo en cuenta (el rastreador y la ubicación de Amazon S3 registrada están en la misma cuenta) - Conceda a Data location (ubicación de los datos) permisos para el rol de IAM que se usa en la ejecución del rastreador en la ubicación de Amazon S3, de modo que el rastreador pueda leer los datos del destino en Lake Formation. Para obtener más información, consulte Concesión de permisos de ubicación de datos (misma cuenta).

Preguntas frecuentes sobre la configuración del rastreador con las credenciales de Lake Formation

1. ¿Cómo configuro un rastreador para que se ejecute con las credenciales de Lake Formation mediante la consola AWS?

   En la consola de AWS Glue (https://console.aws.amazon.com/glue/), al configurar el rastreador, seleccione la opción Use Lake Formation credentials for crawling Amazon S3 data source (Utilizar credenciales de Lake Formation para rastrear un origen de datos de Amazon S3). Para el rastreo entre cuentas, especifique el ID de la Cuenta de AWS donde esté registrada la ubicación de Amazon S3 de destino en Lake Formation. Para realizar el rastreo en cuenta, el campo accountId es opcional.

2. ¿Cómo configuro un rastreador para que se ejecute con las credenciales de Lake Formation mediante AWS CLI?

   Durante la llamada a la API `CreateCrawler`, agregue `LakeFormationConfiguration`:

   ```
   "LakeFormationConfiguration": {
       "UseLakeFormationCredentials": true,
       "AccountId": "111111111111" (AWS account ID where the target Amazon S3 location
    is registered with Lake Formation)
     }
   ```

3. ¿Cuáles son los objetivos admitidos por un rastreador que utiliza las credenciales de Lake Formation?

   Un rastreador que utilice las credenciales de Lake Formation solo es compatible con Amazon S3 (rastreo dentro y entre cuentas) y para los destinos del catálogo de datos de la cuenta (donde la ubicación subyacente es Amazon S3) y destinos Apache Iceberg.

4. ¿Puedo rastrear varios depósitos de Amazon S3 como parte de un único rastreador con las credenciales de Lake Formation?

   No, para los objetivos de rastreo que utilizan la venta de credenciales de Lake Formation, las ubicaciones subyacentes de Amazon S3 deben pertenecer al mismo bucket. Por ejemplo, los clientes pueden utilizar varias ubicaciones de destino (`s3://bucket1/folder1, s3://bucket1/folder2`) si están debajo del mismo bucket (bucket 1). No se admite la especificación de diferentes buckets (s3://bucket1/carpeta1, s3://bucket2/carpeta2).

Solución de problemas AWS Glue de errores de Ray en los registros

AWS Glue proporciona acceso a los registros emitidos por los procesos de Ray durante la ejecución del trabajo. Si encuentra errores o un comportamiento inesperado en los trabajos de Ray, recopile antes información de los registros para determinar la causa del error. También proporcionamos registros similares para sesiones interactivas. Los registros de sesiones se proporcionan con el prefijo `/aws-glue/ray/sessions`.

Las líneas de registro se envían a CloudWatch en tiempo real, a medida que se ejecuta el trabajo. Las declaraciones de impresión se adjuntan a los registros de CloudWatch una vez finalizada la ejecución. Los registros se conservan durante dos semanas después de ejecutar un trabajo.

Inspección de los registros de trabajos de Ray

Cuando se produce un error en un trabajo, recopile el nombre del trabajo y el identificador de ejecución del trabajo. Puede encontrarlos en la consola de AWS Glue. Vaya a la página del trabajo y, a continuación, vaya a la pestaña Runs (Ejecuciones). Los registros de trabajos de Ray se almacenan en los siguientes grupos de registros dedicados de CloudWatch.

- `/aws-glue/ray/jobs/script-log/`: almacena los registros emitidos por su script principal de Ray.
- `/aws-glue/ray/jobs/ray-monitor-log/`: almacena los registros emitidos por el proceso del escalador automático de Ray. Estos registros se generan para el nodo principal y no para otros nodos de trabajo.
- `/aws-glue/ray/jobs/ray-gcs-logs/`: Almacena los registros emitidos por el proceso GCS (almacenamiento de control global). Estos registros se generan para el nodo principal y no para otros nodos de trabajo.
- `/aws-glue/ray/jobs/ray-process-logs/`: Almacena los registros emitidos por otros procesos de Ray (principalmente el agente del panel de control) que se ejecutan en el nodo principal. Estos registros se generan para el nodo principal y no para otros nodos de trabajo.
- `/aws-glue/ray/jobs/ray-raylet-logs/`: Almacena los registros emitidos por cada proceso de raylet. Estos registros se recopilan en una sola secuencia para cada nodo de trabajo, incluido el nodo principal.

- `/aws-glue/ray/jobs/ray-worker-out-logs/`: Almacena los registros `stdout` de cada proceso de trabajo en el clúster. Estos registros se generan para cada nodo de trabajo, incluido el nodo principal.
- `/aws-glue/ray/jobs/ray-worker-err-logs/`: Almacena los registros `stderr` de cada proceso de trabajo en el clúster. Estos registros se generan para cada nodo de trabajo, incluido el nodo principal.
- `/aws-glue/ray/jobs/ray-runtime-env-log/`: Almacena los registros sobre el proceso de configuración de Ray. Estos registros se generan para cada nodo de trabajo, incluido el nodo principal.

Solución de errores de trabajos de Ray

Para entender la organización de los grupos de registro de Ray y encontrar los grupos de registro que ayudarán a solucionar los errores, es útil disponer de información básica sobre la arquitectura de Ray.

En ETL de AWS Glue, un proceso de trabajo corresponde a una instancia. Al configurar procesos de trabajo para un trabajo de AWS Glue, se establece el tipo y la cantidad de instancias dedicadas al trabajo. Ray utiliza el término proceso de trabajo de diferentes maneras.

Ray usa el nodo principal y el nodo de trabajo para distinguir las responsabilidades de una instancia en un clúster de Ray. Un nodo de trabajo de Ray puede alojar procesos de varios actores que hacen cálculos para lograr el resultado de su cálculo distribuido. Los actores que ejecutan una réplica de una función se denominan réplicas. Los actores de réplica también se pueden denominar procesos de trabajo. Las réplicas también se pueden ejecutar en el nodo principal, que se conoce como el principal porque ejecuta procesos adicionales para coordinar el clúster.

Cada actor que contribuye al cálculo genera su propio flujo de registro. Esto nos proporciona un poco de información:

- La cantidad de procesos que emiten registros puede ser mayor que la cantidad de procesos de trabajo asignados al trabajo. A menudo, cada núcleo de cada instancia tiene un actor.
- Los nodos principales de Ray emiten registros de inicio y administración de clústeres. Por el contrario, los nodos de trabajo de Ray solo emiten registros del trabajo hecho en ellos.

Para más información acerca de la arquitectura de Ray, consulte Architecture Whitepapers (Documentos técnicos sobre la arquitectura) en la documentación de Ray.

Área del problema: acceso a Amazon S3

Compruebe el mensaje de error de la ejecución del trabajo. Si eso no proporciona suficiente información, compruebe `/aws-glue/ray/jobs/script-log/`.

Área del problema: administración de dependencias de PIP

Compruebe `/aws-glue/ray/jobs/ray-runtime-env-log/`.

Área del problema: inspección de valores intermedios en el proceso principal

Escriba a `stderr` o `stdout` desde su script principal y recupere los registros de `/aws-glue/ray/jobs/script-log/`.

Área del problema: inspección de valores intermedios en un proceso secundario

Escriba a `stderr` o `stdout` desde su función `remote`. A continuación, recupere los registros de `/aws-glue/ray/jobs/ray-worker-out-logs/` o `/aws-glue/ray/jobs/ray-worker-err-logs/`. Es posible que la función se haya ejecutado en cualquier réplica, por lo que es posible que tenga que examinar varios registros para encontrar el resultado deseado.

Área problemática: interpretación de las direcciones IP en los mensajes de error

En determinadas situaciones de error, es posible que su trabajo emita un mensaje de error que contenga una dirección IP. Estas direcciones IP son efímeras y el clúster las utiliza para identificar los nodos y comunicarse entre ellos. Los registros de un nodo se publicarán en un flujo de registros con un sufijo único basado en la dirección IP.

En CloudWatch, puede filtrar los registros para inspeccionar los específicos de esta dirección IP a través de la identificación de este sufijo. Por ejemplo, si utiliza *`FAILED_IP`* y *`JOB_RUN_ID`*, puede identificar el sufijo con:

```
filter @logStream like /JOB_RUN_ID/
| filter @message like /IP-/
| parse @message "IP-[*]" as ip
| filter ip like /FAILED_IP/
| fields replace(ip, ":", "_") as uIP
| stats count_distinct by uIP as logStreamSuffix
| display logStreamSuffix
```

Excepciones de machine learning en AWS Glue

En este tema se describen los códigos de error HTTP y las cadenas de las excepciones de AWS Glue relacionadas con el machine learning. Se proporcionan los códigos de error y las cadenas de error de cada actividad de machine learning que pueden producirse al realizar una operación. Además, puede ver si es posible volver a intentar la operación que dio lugar al error.

CancelMLTaskRunActivity

Esta actividad tiene las siguientes excepciones:

- EntityNotFoundException (400)
  - "No se puede encontrar MLTransform en la cuenta [AccountId] con el identificador [transformName]".
  - "No se ha encontrado la ML Task Run para [taskRunId]: en la cuenta [accountId] para transformar [transformName]".

  Reintentar: No.

CreateMLTaskRunActivity

Esta actividad tiene las siguientes excepciones:

- InvalidInputException (400)
  - "Error de servicio interno debido a una entrada inesperada".
  - "Se debe especificar un origen de entrada de la tabla de AWS Glue en la transformación".
  - "La columna de origen de entrada [columnName] tiene un tipo de datos no válido definido en el catálogo".
  - "Se debe proporcionar exactamente una tabla de registro de entrada".
  - "Se debe especificar el nombre de la base de datos".
  - "Se debe especificar el nombre de la tabla".
  - "El esquema no está definido en la transformación".
  - "El esquema debe contener la clave principal proporcionada: [primaryKey]".
  - "Problema al recuperar el esquema del catálogo de datos: [message]".
  - "No se puede establecer capacidad máxima y número de trabajo/tipo al mismo tiempo".

- "Se deben establecer tanto WorkerType como NumberOfWorkers".
- "MaxCapacity debe ser >= [maxCapacity]".
- "NumberOfWorkers debe ser >= [maxCapacity]".
- "Los reintentos máximos no deben ser negativos".
- "No se han establecido los parámetros de búsqueda de coincidencias".
- "Se debe especificar una clave principal en los parámetros de búsqueda de coincidencias".

Reintentar: No.

- AlreadyExistsException (400)
  - "Ya existe una transformación con el nombre [transformName]".

Reintentar: No.

- IdempotentParameterMismatchException (400)
  - "La solicitud de creación de Idempotent para la transformación [transformName] tenía parámetros que no coincidían".

Reintentar: No.

- InternalServiceException (500)
  - "Error de dependencia".

Reintentar: Sí.

- ResourceNumberLimitExceededException (400)
  - "El recuento de Transformaciones ML ([count]) ha excedido el límite de [limit] transformaciones".

Reintentar: Sí, una vez que haya eliminado una transformación para dejar espacio a esta nueva.

DeleteMLTransformActivity

Esta actividad tiene las siguientes excepciones:

- EntityNotFoundException (400)
  - "No se puede encontrar MLTransform en la cuenta [accountId] con el identificador [transformName]"

Reintentar: No.

GetMLTaskRunActivity

Esta actividad tiene las siguientes excepciones:

- EntityNotFoundException (400)
  - "No se puede encontrar MLTransform en la cuenta [AccountId] con el identificador [transformName]".
  - "No se ha encontrado la ML Task Run para [taskRunId]: en la cuenta [accountId] para transformar [transformName]".

  Reintentar: No.

GetMLTaskRunsActivity

Esta actividad tiene las siguientes excepciones:

- EntityNotFoundException (400)
  - "No se puede encontrar MLTransform en la cuenta [AccountId] con el identificador [transformName]".
  - "No se ha encontrado la ML Task Run para [taskRunId]: en la cuenta [accountId] para transformar [transformName]".

  Reintentar: No.

GetMLTransformActivity

Esta actividad tiene las siguientes excepciones:

- EntityNotFoundException (400)
  - "No se puede encontrar MLTransform en la cuenta [AccountId] con el identificador [transformName]".

  Reintentar: No.

GetMLTransformsActivity

Esta actividad tiene las siguientes excepciones:

- EntityNotFoundException (400)
  - "No se puede encontrar MLTransform en la cuenta [AccountId] con el identificador [transformName]".

  Reintentar: No.
- InvalidInputException (400)
  - "El ID de cuenta no puede estar en blanco".
  - "No se admite clasificación para la columna [column]".
  - "[column] no puede estar en blanco".
  - "Error de servicio interno debido a una entrada inesperada".

  Reintentar: No.

GetSaveLocationForTransformArtifactActivity

Esta actividad tiene las siguientes excepciones:

- EntityNotFoundException (400)
  - "No se puede encontrar MLTransform en la cuenta [AccountId] con el identificador [transformName]".

  Reintentar: No.
- InvalidInputException (400)
  - "Tipo de artefacto no admitido [ArtifactType]".
  - "Error de servicio interno debido a una entrada inesperada".

  Reintentar: No.

GetTaskRunArtifactActivity

Esta actividad tiene las siguientes excepciones:

- EntityNotFoundException (400)
  - "No se puede encontrar MLTransform en la cuenta [AccountId] con el identificador [transformName]".

- "No se ha encontrado la ML Task Run para [taskRunId]: en la cuenta [accountId] para transformar [transformName]".

Reintentar: No.

- InvalidInputException (400)
  - "El nombre de archivo '[fileName]' no es válido para la publicación".
  - "No se puede recuperar el artefacto para el tipo de tarea [taskType]".
  - "No se puede recuperar artefacto para [artifactType]".
  - "Error de servicio interno debido a una entrada inesperada".

Reintentar: No.

PublishMLTransformModelActivity

Esta actividad tiene las siguientes excepciones:

- EntityNotFoundException (400)
  - "No se puede encontrar MLTransform en la cuenta [AccountId] con el identificador [transformName]".
  - "No se puede encontrar un modelo existente con la versión: [version] para el id de cuenta: [accountId] y el id de transformación - [transformId]".

Reintentar: No.

- InvalidInputException (400)
  - "El nombre de archivo '[fileName]' no es válido para la publicación".
  - "Signo menos inicial ilegal en cadena sin signo [string]".
  - "Dígito incorrecto al final de [string]".
  - "El valor de la cadena [string] excede el rango de longitud sin signo".
  - "Error de servicio interno debido a una entrada inesperada".

Reintentar: No.

PullLatestMLTransformModelActivity

Esta actividad tiene las siguientes excepciones:

- EntityNotFoundException (400)
  - "No se puede encontrar MLTransform en la cuenta [AccountId] con el identificador [transformName]".

  Reintentar: No.
- InvalidInputException (400)
  - "Error de servicio interno debido a una entrada inesperada".

  Reintentar: No.
- ConcurrentModificationException (400)
  - "No se puede crear la versión del modelo que se quiere capacitar debido a inserciones de carreras con parámetros que no coinciden".
  - "El modelo de transformación ML para el id de transformación [transformId] está obsoleto o está siendo actualizado por otro proceso; vuelva a intentarlo".

  Reintentar: Sí.

PutJobMetadataForMLTransformActivity

Esta actividad tiene las siguientes excepciones:

- EntityNotFoundException (400)
  - "No se puede encontrar MLTransform en la cuenta [AccountId] con el identificador [transformName]".
  - "No se ha encontrado la ML Task Run para [taskRunId]: en la cuenta [accountId] para transformar [transformName]".

  Reintentar: No.
- InvalidInputException (400)
  - "Error de servicio interno debido a una entrada inesperada".
  - "Tipo de metadatos de trabajo desconocido [jobType]".
  - "Debe proporcionar un ID de ejecución de tarea para actualizar".

  Reintentar: No.

StartExportLabelsTaskRunActivity

Esta actividad tiene las siguientes excepciones:

- EntityNotFoundException (400)
  - "No se puede encontrar MLTransform en la cuenta [AccountId] con el identificador [transformName]".
  - "No existe ningún conjunto de etiquetas para transformId [transformID] en el ID de cuenta [accountID]".

  Reintentar: No.
- InvalidInputException (400)
  - “[message].”
  - "La ruta S3 proporcionada no está en la misma región que la transformación. Se esperaba región: [región], pero se obtuvo: [región]".

  Reintentar: No.

StartImportLabelsTaskRunActivity

Esta actividad tiene las siguientes excepciones:

- EntityNotFoundException (400)
  - "No se puede encontrar MLTransform en la cuenta [AccountId] con el identificador [transformName]".

  Reintentar: No.
- InvalidInputException (400)
  - “[message].”
  - "Ruta de archivo de etiqueta no válida".
  - "No se puede acceder al archivo de etiqueta en [labelPath]. [message]".
  - "No se puede utilizar el rol de IAM proporcionado en la transformación. Rol: [role]".
  - "Archivo de etiqueta de tamaño 0 no válido".
  - "La ruta S3 proporcionada no está en la misma región que la transformación. Se esperaba región: [región], pero se obtuvo: [región]".

Reintentar: No.

- ResourceNumberLimitExceededException (400)
  - "El archivo de etiqueta ha superado el límite de [limit] MB".

  Reintentar: No. Considere dividir su archivo de etiqueta en varios archivos más pequeños.

StartMLEvaluationTaskRunActivity

Esta actividad tiene las siguientes excepciones:

- EntityNotFoundException (400)
  - "No se puede encontrar MLTransform en la cuenta [AccountId] con el identificador [transformName]".

  Reintentar: No.
- InvalidInputException (400)
  - "Se debe proporcionar exactamente una tabla de registro de entrada".
  - "Se debe especificar el nombre de la base de datos".
  - "Se debe especificar el nombre de la tabla".
  - "No se han establecido los parámetros de búsqueda de coincidencias".
  - "Se debe especificar una clave principal en los parámetros de búsqueda de coincidencias".

  Reintentar: No.
- MLTransformNotReadyException (400)
  - "Esta operación solo se puede aplicar a una transformación que se encuentra en un estado READY".

  Reintentar: No.
- InternalServiceException (500)
  - "Error de dependencia".

  Reintentar: Sí.
- ConcurrentRunsExceededException (400)
  - "El recuento de ML Task Runs [count] ha superado el límite de transformación de las ejecuciones de [limit] tareas".

- "El recuento de ML Task Runs [count] ha superado el límite de ejecuciones de [limit] tareas".

Reintentar: Sí, después de esperar a que finalicen las ejecuciones de tareas.

StartMLLabelingSetGenerationTaskRunActivity

Esta actividad tiene las siguientes excepciones:

- EntityNotFoundException (400)
  - "No se puede encontrar MLTransform en la cuenta [AccountId] con el identificador [transformName]".

  Reintentar: No.
- InvalidInputException (400)
  - "Se debe proporcionar exactamente una tabla de registro de entrada".
  - "Se debe especificar el nombre de la base de datos".
  - "Se debe especificar el nombre de la tabla".
  - "No se han establecido los parámetros de búsqueda de coincidencias".
  - "Se debe especificar una clave principal en los parámetros de búsqueda de coincidencias".

  Reintentar: No.
- InternalServiceException (500)
  - "Error de dependencia".

  Reintentar: Sí.
- ConcurrentRunsExceededException (400)
  - "El recuento de ML Task Runs [count] ha superado el límite de transformación de las ejecuciones de [limit] tareas".

  Reintentar: Sí, después de que se hayan completado las ejecuciones de tareas.

UpdateMLTransformActivity

Esta actividad tiene las siguientes excepciones:

- EntityNotFoundException (400)

- "No se puede encontrar MLTransform en la cuenta [AccountId] con el identificador [transformName]".

Reintentar: No.

- InvalidInputException (400)
  - "Ya existe otra transformación con el nombre [transformName]".
  - “[message].”
  - "El nombre de la transformación no puede estar en blanco".
  - "No se puede establecer capacidad máxima y número de trabajo/tipo al mismo tiempo".
  - "Se deben establecer tanto WorkerType como NumberOfWorkers".
  - "MaxCapacity debe ser >= [minMaxCapacity]".
  - "NumberOfWorkers debe ser >= [minNumWorkers]".
  - "Los reintentos máximos no deben ser negativos".
  - "Error de servicio interno debido a una entrada inesperada".
  - "No se han establecido los parámetros de búsqueda de coincidencias".
  - "Se debe especificar una clave principal en los parámetros de búsqueda de coincidencias".

  Reintentar: No.
- AlreadyExistsException (400)
  - "Ya existe una transformación con el nombre [transformName]".

  Reintentar: No.
- IdempotentParameterMismatchException (400)
  - "La solicitud de creación de Idempotent para la transformación [transformName] tenía parámetros que no coincidían".

  Reintentar: No.

Cuotas de AWS Glue

Puede ponerse en contacto con AWS Supportpara solicitar un aumento de cuota para las service quotas indicadas en el Referencia general de AWS. A menos que se indique lo contrario, cada cuota es específica de la región de . Para obtener más información, consulte Puntos de conexión y cuotas de AWS Glue.

Mejora del AWS Glue rendimiento

Estrategia básica para ajustar el rendimiento

Para mejorar el AWS Glue rendimiento, puede considerar la posibilidad de actualizar determinados AWS Glue parámetros relacionados con el rendimiento. Cuando se prepare para ajustar los parámetros, siga las siguientes prácticas recomendadas:

- Determine sus objetivos de rendimiento antes de empezar a identificar los problemas.
- Utilice las métricas para identificar los problemas antes de intentar cambiar los parámetros de ajuste.

Para obtener resultados más consistentes al afinar un trabajo, desarrolle una estrategia básica para su trabajo de ajuste.

Por lo general, el ajuste del rendimiento se realiza en el siguiente flujo de trabajo:

1. Determine los objetivos de rendimiento.
2. Mida las métricas.
3. Identifique los cuellos de botella.
4. Reduzca el impacto de los cuellos de botella.
5. Repita los pasos 2 a 4 hasta alcanzar el objetivo deseado.

Ajuste de las estrategias para el tipo de trabajo

Trabajos de Spark: siga las instrucciones de prácticas recomendadas AWS Glue para ajustar el rendimiento de los trabajos de Apache Spark en AWS Prescriptive Guidance.

Otros trabajos: puede ajustar los trabajos AWS Glue de shell de Ray y AWS Glue Python adaptando las estrategias disponibles en otros entornos de ejecución.

Mejora del rendimiento de AWS Glue para los trabajos de Apache Spark

Para mejorar el rendimiento de AWS Glue para Spark, le recomendamos actualizar algunos parámetros relacionados con el rendimiento de AWS Glue y de Spark.

Para obtener más información sobre estrategias específicas para identificar los cuellos de botella mediante métricas y reducir su impacto, consulte las Prácticas recomendadas para el rendimiento al ajustar AWS Glue para los trabajos de Apache Spark en Recomendaciones de AWS. Esta guía presenta los temas principales que se pueden aplicar a Apache Spark en todos los entornos de tiempo de ejecución, como la arquitectura de Spark y los conjuntos de datos distribuidos resilientes. Con estos temas, la guía ayuda para implementar estrategias específicas de ajuste del rendimiento, como la optimización de las combinaciones y la paralelización de las tareas.

Se pueden identificar los cuellos de botella al configurar AWS Glue para que muestre la interfaz de usuario de Spark. Para obtener más información, consulte the section called “Monitorización con la interfaz de usuario de Spark”.

Además, AWS Glue ofrece características de rendimiento que pueden aplicarse al tipo específico de almacén de datos al que se conecta el trabajo. Puede encontrar información de referencia sobre los parámetros de rendimiento para los almacenes de datos en the section called “Parámetros de conexión”.

Optimización de las lecturas con inserción en operaciones de ETL de AWS Glue

El método Pushdown es una técnica de optimización que acerca la lógica de la recuperación de datos al origen de los datos. El origen podría ser una base de datos o un sistema de archivos como Amazon S3. Al ejecutar determinadas operaciones directamente en el origen, puede ahorrar tiempo y potencia de procesamiento al no llevar todos los datos de la red al motor Spark gestionado por Glue AWS.

Otra forma de decir esto es que la presión hacia abajo reduce el escaneo de datos. Para obtener más información sobre el proceso de identificación de cuándo esta técnica es adecuada, consulte Reducir la cantidad de datos escaneados en la guía de Prácticas recomendadas para el ajuste del rendimiento de AWS Glue para trabajos de Apache Spark en las Recomendaciones de AWS.

Predica cómo insertar los archivos almacenados en Amazon S3

Al trabajar con archivos en Amazon S3 organizados por prefijo, puede filtrar las rutas de Amazon S3 de destino mediante la definición de un predicado de inserción. En lugar de leer el conjunto de datos completo y aplicar filtros dentro de un `DynamicFrame`, puede aplicar el filtro directamente a los metadatos de la partición almacenados en el catálogo de datos de Glue AWS. Este enfoque

permite enumerar y leer de forma selectiva solo los datos necesarios. Para obtener más información sobre este proceso, incluida la escritura en un bucket mediante particiones, consulte the section called “Administración de particiones”.

Para lograr la inserción de predicados en Amazon S3, utilice el parámetro `push_down_predicate`. Pensemos en un bucket de Amazon S3 que haya particionado por año, mes y día. Si quiere recuperar los datos de los clientes de junio de 2022, puede indicar a Glue AWS que lea solo las rutas relevantes de Amazon S3. En este caso, el `push_down_predicate` es `year='2022' and month='06'`. Juntándolo todo, la operación de lectura se puede lograr de la siguiente manera:

Python

```
customer_records = glueContext.create_dynamic_frame.from_catalog(
    database = "customer_db",
    table_name = "customer_tbl",
    push_down_predicate = "year='2022' and month='06'"
)
```

Scala

```
val customer_records = glueContext.getCatalogSource(
database="customer_db",
tableName="customer_tbl",
pushDownPredicate="year='2022' and month='06'"
).getDynamicFrame()
```

En el escenario anterior, `push_down_predicate` recupera una lista de todas las particiones del catálogo de datos de Glue AWS y las filtra antes de leer los archivos subyacentes de Amazon S3. Aunque esto ayuda en la mayoría de los casos, cuando se trabaja con conjuntos de datos que tienen millones de particiones, el proceso de enumerar las particiones puede llevar mucho tiempo. Para solucionar este problema, se puede utilizar la depuración de las particiones desde el servidor para mejorar el rendimiento. Para ello, se crea un índice de particiones para los datos en el catálogo de datos de Glue AWS. Para obtener más información sobre los índices de particiones, consulte the section called “Trabajar con índices de partición”. A continuación, puede utilizar la opción `catalogPartitionPredicate` para hacer referencia al índice. Para ver un ejemplo de cómo recuperar particiones con `catalogPartitionPredicate`, consulte the section called “Predicados de particiones de catálogo”.

Insertar al trabajar con fuentes JDBC

El lector JDBC de Glue AWS que se utiliza en el archivo `GlueContext` admite la inserción en las bases de datos compatibles al proporcionar consultas SQL personalizadas que se pueden ejecutar directamente en el origen. Esto se puede lograr mediante la configuración del parámetro `sampleQuery`. La consulta de ejemplo puede especificar qué columnas seleccionar, así como proporcionar un predicado de inserción para limitar los datos transferidos al motor Spark.

De forma predeterminada, las consultas de muestra funcionan en un único nodo, lo que puede provocar fallos en el trabajo cuando se manejan grandes volúmenes de datos. Para utilizar esta función para consultar datos a escala, debe configurar la partición de consultas con el valor `enablePartitioningForSampleQuery` a verdadero, lo que distribuirá la consulta a varios nodos en la clave que elija. La partición de consultas también requiere algunos otros parámetros de configuración necesarios. Para obtener más información sobre la partición de consultas, vea the section called “Lectura desde JDBC en paralelo”.

Al configurar `enablePartitioningForSampleQuery`, Glue AWS combinará el predicado de inserción con un predicado de partición al consultar la base de datos. Su `sampleQuery` debe terminar con un `AND` para que Glue AWS agregue condiciones de partición. (Si no proporciona un predicado de inserción, `sampleQuery` debe terminar con un `WHERE`). Consulte un ejemplo a continuación, en el que presionamos un predicado hacia abajo para recuperar solo las `id` filas superiores a 1000. Esto `sampleQuery` solo devolverá las columnas de nombre y ubicación de las filas donde `id` sea mayor que el valor especificado:

Python

```
sample_query = "select name, location from customer_tbl WHERE id>=1000 AND"
customer_records = glueContext.create_dynamic_frame.from_catalog(
    database="customer_db",
    table_name="customer_tbl",
    sample_query = "select name, location from customer_tbl WHERE id>=1000 AND",

    additional_options = {
                           "hashpartitions": 36 ,
                           "hashfield":"id",
                           "enablePartitioningForSampleQuery":True,
                           "sampleQuery":sample_query
                          }
)
```

Scala

```
val additionalOptions = Map(
        "hashpartitions" -> "36",
        "hashfield" -> "id",
        "enablePartitioningForSampleQuery" -> "true",
        "sampleQuery" -> "select name, location from customer_tbl WHERE id >= 1000
 AND"
        )

    val customer_records = glueContext.getCatalogSource(
        database="customer_db",
        tableName="customer_tbl").getDynamicFrame()
```

> **Note**
>
> Si el nombre de `customer_tbl` es diferente en el Catálogo de datos en el almacén de datos subyacente, debe brindar el nombre subyacente de la tabla en sample_query, ya que la solicitud es enviada al almacén de datos subyacente.

También puede realizar consultas en tablas JDBC sin necesidad de integrarlas con el catálogo de datos de Glue AWS. En lugar de proporcionar el nombre de usuario y la contraseña como parámetros del método, puede reutilizar las credenciales de una conexión preexistente y proporcionar `useConnectionProperties` y `connectionName`. En este ejemplo, recuperamos las credenciales de una conexión llamada `my_postgre_connection`.

Python

```
connection_options_dict = {
    "useConnectionProperties": True,
    "connectionName": "my_postgre_connection",
    "dbtable":"customer_tbl",
    "sampleQuery":"select name, location from customer_tbl WHERE id>=1000 AND",
    "enablePartitioningForSampleQuery":True,
    "hashfield":"id",
    "hashpartitions":36
    }
```

```
customer_records = glueContext.create_dynamic_frame.from_options(
    connection_type="postgresql",
    connection_options=connection_options_dict
    )
```

Scala

```
val connectionOptionsJson = """
      {
        "useConnectionProperties": true,
        "connectionName": "my_postgre_connection",
        "dbtable": "customer_tbl",
        "sampleQuery": "select name, location from customer_tbl WHERE id>=1000 AND",
        "enablePartitioningForSampleQuery" : true,
        "hashfield" : "id",
        "hashpartitions" : 36
      }
    """

    val connectionOptions = new JsonOptions(connectionOptionsJson)

    val dyf = glueContext.getSource("postgresql",
 connectionOptions).getDynamicFrame()
```

Notas y limitaciones de cómo insertar en Glue AWS

El concepto Pushdown es aplicable cuando se lee desde orígenes que no son de streaming. AWS Glue admite una variedad de orígenes: la capacidad de insertar depende de la fuente y el conector.

- Cuando se conecte a Snowflake, puede usar la opción `query`. Existe una funcionalidad similar en el conector Redshift de Glue 4.0 AWS y versiones posteriores. Para obtener más información acerca de cómo leer de Snowflake with`query`, consulte the section called “Leer de Snowflake”.
- El lector de ETL de DynamoDB no es compatible con filtros ni predicados de inserción. MongoDB y DocumentDB tampoco admiten este tipo de funcionalidad.
- Al leer datos almacenados en Amazon S3 en formatos de tabla abierta, el método de partición de los archivos en Amazon S3 ya no es suficiente. Para leer y escribir desde particiones que utilizan formatos de tabla abierta, consulte la documentación del formato.
- Los métodos DynamicFrame no realizan la inserción de proyección de Amazon S3. Se leerán todas las columnas de los archivos que pasen el filtro de predicados.

- Cuando se trabaja con conectores `custom.jdbc` en Glue AWS, la capacidad de inserción depende del origen y del conector. Revise la documentación del conector correspondiente para confirmar si es compatible con la inserción en Glue AWS y cómo lo hace.

Utilizar Auto Scaling para AWS Glue

Auto Scaling está disponible para trabajos de ETL y streaming de AWS Glue con AWS Glue versión 3.0 o posterior.

Con Auto Scaling habilitado, obtendrá los siguientes beneficios:

- AWS Glue agrega automáticamente y elimina empleados del clúster en función del paralelismo en cada etapa o microlote de la ejecución del trabajo.
- Elimina la necesidad de experimentar y decidir el número de empleados que asignará a sus trabajos de ETL de AWS Glue.
- Si elije el número máximo de empleados, AWS Glue elegirá el tamaño correcto de recursos para la carga de trabajo.
- Puede ver cómo cambia el tamaño del clúster durante la ejecución del trabajo si mira las métricas de CloudWatch en la página de detalles de ejecución de trabajo de AWS Glue Studio.

Auto Scaling para trabajos de ETL de AWS Glue y streaming permite escalado vertical y reducción vertical bajo demanda de los recursos informáticos de sus trabajos de AWS Glue. El escalado vertical bajo demanda lo ayuda a asignar solo los recursos informáticos necesarios, en principio, en el inicio de la ejecución de trabajos, así como a aprovisionar los recursos necesarios según la demanda durante el trabajo.

Auto Scaling también admite la reducción vertical dinámica de los recursos de trabajo de AWS Glue durante el transcurso de un trabajo. Durante la ejecución de un trabajo, cuando la aplicación Spark solicite más ejecutores, se agregarán más empleados al clúster. Cuando el ejecutor ha estado inactivo sin tareas de cálculo activas, se eliminarán el ejecutor y el empleado correspondiente.

Los escenarios comunes en los que Auto Scaling ayuda con el costo y la utilización de sus aplicaciones Spark incluyen un controlador de Spark que enumera un gran número de archivos en Amazon S3 o realiza una carga mientras los ejecutores se encuentran inactivos, etapas en las que Spark se ejecuta con solo unos pocos ejecutores debido al sobreaprovisionamiento y sesgos de datos o demanda de cálculo desigual en todas las etapas de Spark.

Requisitos

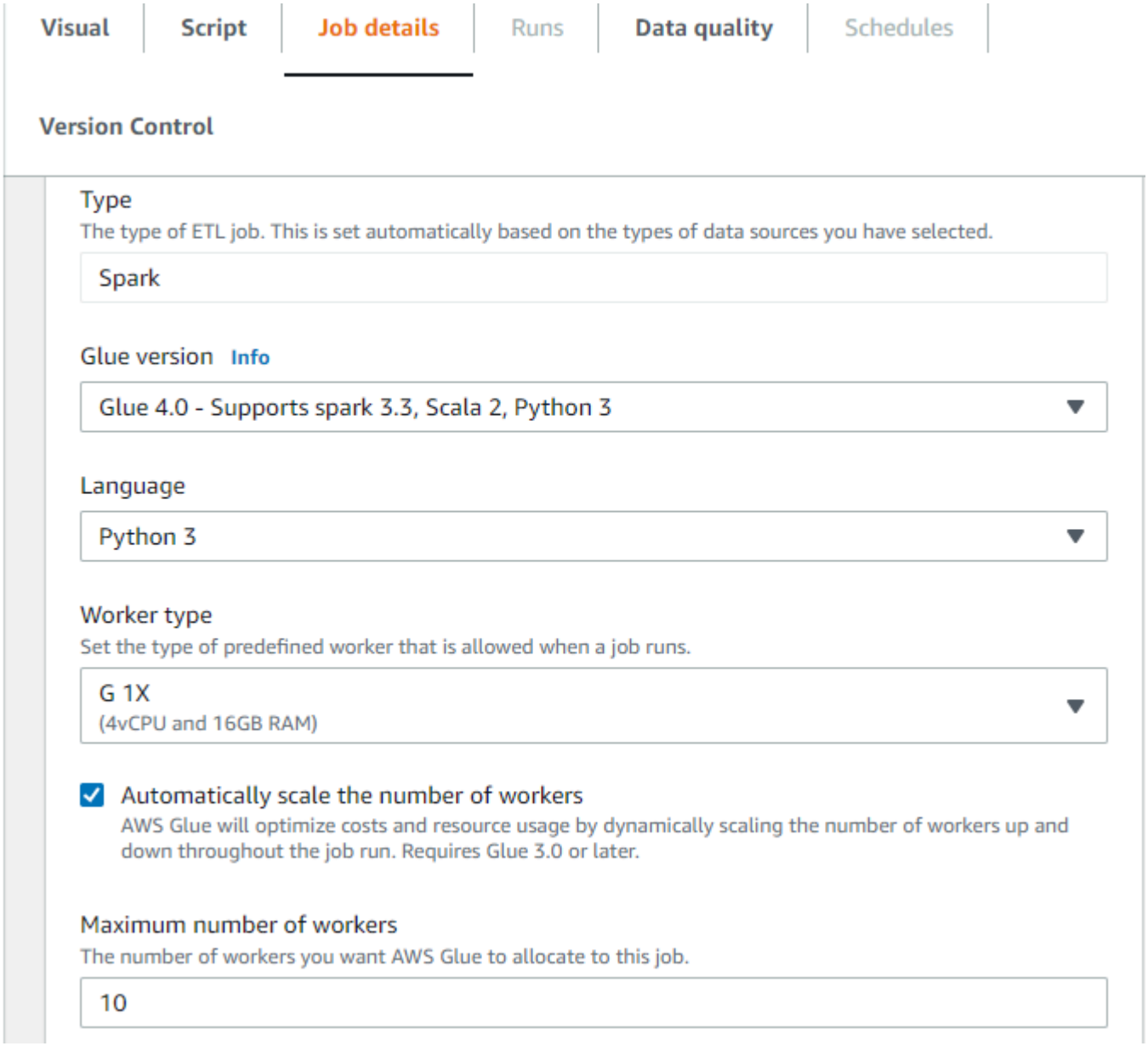
El escalado automático solo se encuentra disponible para la versión 3.0 o posterior de AWS Glue. Para utilizar el escalado automático, puede seguir la guía de migración a fin de migrar los trabajos existentes a la versión 3.0 o posterior de AWS Glue, o bien crear trabajos nuevos con la versión 3.0 o posterior de AWS Glue.

El escalado automático se encuentra disponible para trabajos de AWS Glue con los tipos de trabajador `G.1X`, `G.2X`, `G.4X`, `G.8X` o `G.025X` (solo con trabajos de streaming). No se admiten las DPU estándar.

Habilitar Auto Scaling en AWS Glue Studio

En la página Job details (Detalles del trabajo) en AWS Glue Studio, elija el tipo como Spark o Spark Streaming y Glue version (Versión de Glue) como **`Glue 3.0`** o **`Glue 4.0`**. A continuación, se mostrará una casilla de verificación debajo de Worker type (Tipo de empleado).

- Seleccione la opción Automatically scale the number of workers (Escalar automáticamente el número de empleados).
- Establezca el Maximum number of workers (Número máximo de empleados) para definir el número máximo de empleados que se pueden asignar a la ejecución de trabajos.



Habilitación de Auto Scaling con AWS CLI o SDK

Para habilitar Auto Scaling desde la AWS CLI para la ejecución de un trabajo, ejecute `start-job-run` con la siguiente configuración:

```
{
    "JobName": "<your job name>",
    "Arguments": {
        "--enable-auto-scaling": "true"
    },
    "WorkerType": "G.2X", // G.1X and G.2X are allowed for Auto Scaling Jobs
```

```
    "NumberOfWorkers": 20, // represents Maximum number of workers
    ...other job run configurations...
}
```

Una vez que haya finalizado la ejecución de un trabajo de ETL, también puede llamar a `get-job-run` para comprobar el uso real de recursos de la ejecución del trabajo en segundos de DPU. Nota: El nuevo campo DPUSeconds solo se mostrará para los trabajos por lotes en AWS Glue 3.0 o posterior con el escalado automático habilitado. Este campo no es compatible con trabajos de streaming.

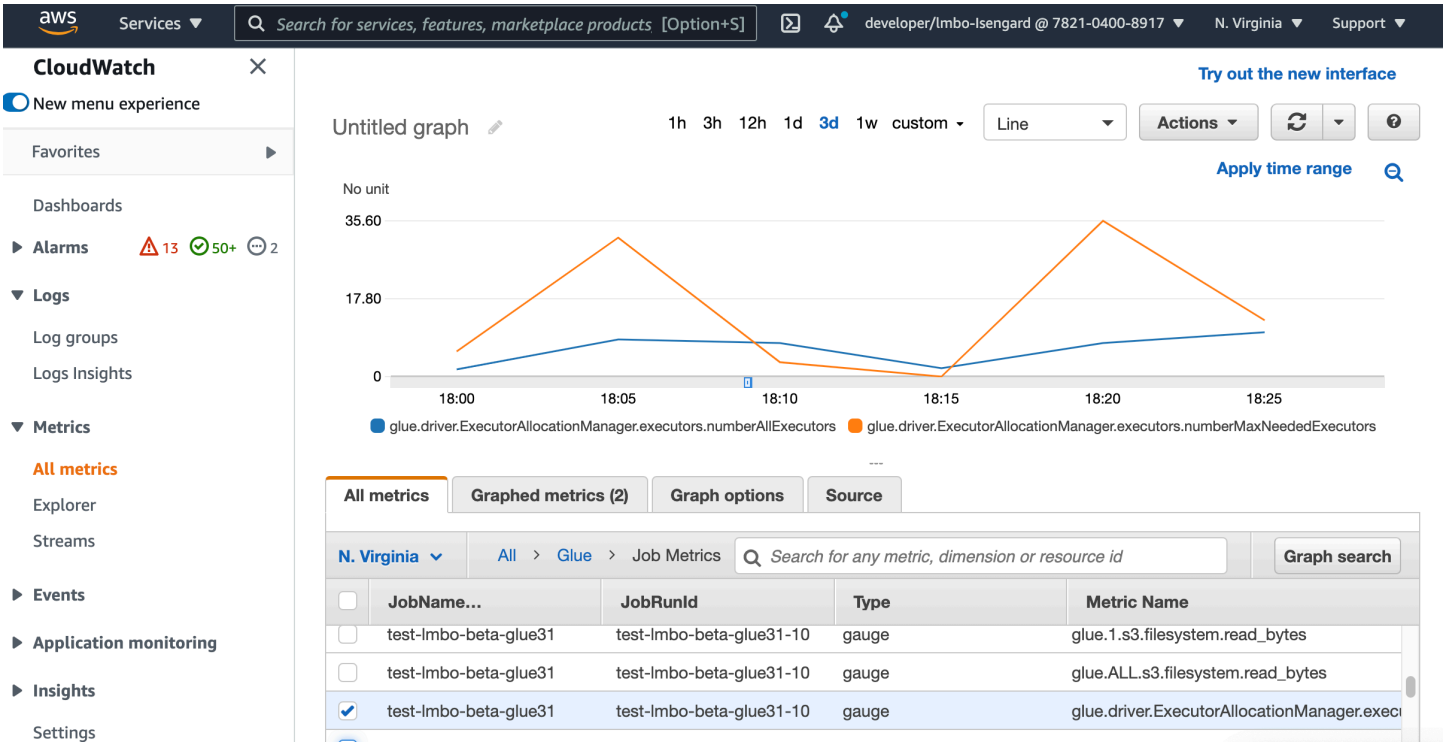
```
$ aws glue get-job-run --job-name your-job-name --run-id jr_xx --endpoint https://
glue.us-east-1.amazonaws.com --region us-east-1
{
    "JobRun": {
        ...
        "GlueVersion": "3.0",
        "DPUSeconds": 386.0
    }
}
```

También se pueden configurar ejecuciones de trabajos con Auto Scaling mediante el AWS Glue SDK con la misma configuración.

Monitorear Auto Scaling con las métricas de Amazon CloudWatch.

Las métricas de ejecutor de CloudWatch se encuentran disponibles para sus trabajos de AWS Glue 3.0 o posterior si habilita el escalado automático. Las métricas se pueden utilizar para monitorear la demanda y la utilización optimizada de los ejecutores en sus aplicaciones Spark habilitadas con Auto Scaling. Para obtener más información, consulte Supervisión de AWS Glue con métricas de Amazon CloudWatch.

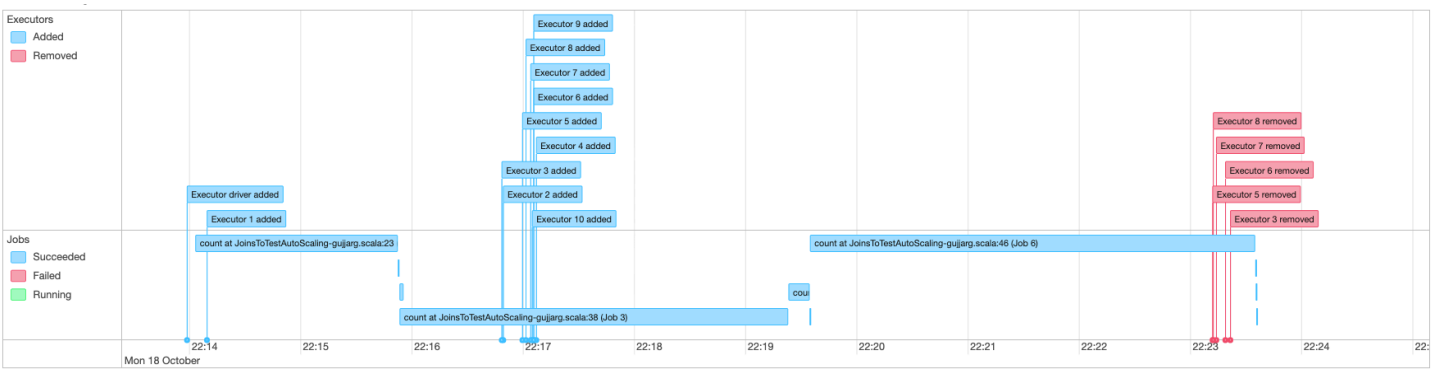
- glue.driver.ExecutorAllocationManager.executors.numberAllExecutors
- glue.driver.ExecutorAllocationManager.executors.numberMaxNeededExecutors



Para obtener más información sobre estas métricas, consulte Monitorización de la planificación de la capacidad de DPU.

Monitorear Auto Scaling con la IU de Spark

Con Auto Scaling habilitado, también puede supervisar los ejecutores que se agregan y eliminan con escalado vertical y reducción vertical dinámicos en función de la demanda de sus trabajos de AWS Glue mediante la IU de Glue Spark. Para obtener más información, consulte Habilitación de la interfaz de usuario web de Apache Spark para trabajos de AWS Glue.



Supervisión del uso de DPU en ejecución de trabajos de Auto Scaling

Puede utilizar la Vista de ejecución de trabajos de AWS Glue Studio para comprobar el uso de DPU de los trabajos de escalado automático.

1. Elija Supervisión en el panel de navegación de AWS Glue Studio. Aparecerá la página Monitoring (Supervisión).
2. Desplácese hacia abajo hasta el gráfico Job runs (Ejecuciones de trabajos).
3. Navegue hasta la ejecución del trabajo que le interese y desplácese hasta la columna DPU hours (Horas de DPU) para comprobar el uso de la ejecución del trabajo en cuestión.

Limitaciones

En la actualidad, el streaming de AWS Glue de Auto Scaling no soporta una unión de DataFrame en streaming con un DataFrame estático creado fuera de `ForEachBatch`. Un DataFrame estático creado dentro de `ForEachBatch` funcionará según lo previsto.

Partición de cargas de trabajo con ejecución limitada

Los errores en las aplicaciones de Spark suelen deberse a scripts de Spark ineficientes, a la ejecución distribuida en memoria de transformaciones a gran escala y a anomalías de los conjuntos de datos. Hay muchas razones que pueden causar problemas de memoria al controlador o al ejecutor, por ejemplo, un sesgo de datos, una lista con demasiados objetos o mezclas aleatorias importantes de datos. Estos problemas suelen aparecer cuando se procesan grandes cantidades de datos pendientes con Spark.

AWS Glue le permite resolver problemas de memoria insuficiente y facilitar el procesamiento de ETL con la partición de carga de trabajo. Si habilita la partición de carga de trabajo, cada ejecución de trabajo de ETL solo selecciona datos no procesados, con un límite superior en el tamaño del conjunto de datos o la cantidad de archivos que se van a procesar con esta ejecución de trabajo. Las ejecuciones de trabajos futuras procesarán los datos restantes. Por ejemplo, si hay 1000 archivos que necesitan ser procesados, puede establecer el número de archivos en 500 y separarlos en dos ejecuciones de trabajos.

La partición de carga de trabajo sólo se admite para orígenes de datos de Amazon S3.

Habilitar la partición de carga de trabajo

Puede habilitar la ejecución limitada mediante la configuración manual de las opciones en el script o mediante el agregado de propiedades a la tabla del catálogo.

Para habilitar la partición de carga de trabajo con ejecución limitada en el script:

1. Para evitar el reprocesamiento de datos, habilite los marcadores de trabajos en el trabajo nuevo o en el trabajo existente. Para obtener más información, consulte Seguimiento de los datos procesados mediante marcadores de trabajo.
2. Modifique el script y establezca el límite delimitado en las opciones adicionales de la API `getSource` de AWS Glue. También debe establecer el contexto de transformación para que el marcador de trabajo almacene el elemento `state`. Por ejemplo:

   Python

   ```
   glueContext.create_dynamic_frame.from_catalog(
       database = "database",
       table_name = "table_name",
       redshift_tmp_dir = "",
       transformation_ctx = "datasource0",
       additional_options = {
           "boundedFiles" : "500", # need to be string
         # "boundedSize" : "1000000000" unit is byte
       }
   )
   ```

   Scala

   ```
   val datasource0 = glueContext.getCatalogSource(
       database = "database", tableName = "table_name", redshiftTmpDir = "",
       transformationContext = "datasource0",
       additionalOptions = JsonOptions(
           Map("boundedFiles" -> "500") // need to be string
         //"boundedSize" -> "1000000000" unit is byte
       )
   ).getDynamicFrame()
   ```

   ```
   val connectionOptions = JsonOptions(
       Map("paths" -> List(baseLocation), "boundedFiles" -> "30")
   ```

```
)
val source = glueContext.getSource("s3", connectionOptions, "datasource0", "")
```

Para habilitar la partición de carga de trabajo con ejecución limitada en la tabla de Data Catalog:

1. Configure los pares clave-valor en el campo `parameters` de la estructura de su tabla en Data Catalog. Para obtener más información, consulte Visualización y edición de los detalles de la tabla.
2. Configure el límite superior para el tamaño del conjunto de datos o la cantidad de archivos procesados:

   - Configure el `boundedSize` al tamaño objetivo del conjunto de datos en bytes. La ejecución del trabajo se detendrá después de alcanzar el tamaño objetivo de la tabla.
   - Configure `boundedFiles` a la cantidad de archivos objetivo. La ejecución del trabajo se detendrá después de procesar la cantidad de archivos objetivo.

> **Note**
>
> Solo debe configurar uno entre `boundedSize` o `boundedFiles`, ya que solo se soporta un único límite.

Configuración de un desencadenador de AWS Glue para ejecutar el trabajo en forma automática

Una vez que haya habilitado la ejecución limitada, puede configurar un desencadenador de AWS Glue para ejecutar el trabajo en forma automática y cargar los datos de forma progresiva en ejecuciones secuenciales. Diríjase a la consola de AWS Glue y cree un desencadenador, configure la hora programada y adjúntelo a su trabajo. Se desencadenará automáticamente la siguiente ejecución del trabajo y se procesará el nuevo lote de datos.

También puede utilizar los flujos de trabajo de AWS Glue u orquestar múltiples trabajos para procesar datos de diferentes particiones en paralelo. Para obtener más información, consulte Desencadenadores de AWS Glue y Flujos de trabajo de AWS Glue.

Para obtener más información sobre los casos de uso y las opciones, consulte el blog Optimizing Spark applications with workload partitioning in AWS Glue (Optimización de aplicaciones de Spark con partición de carga de trabajo en Glue).

Problemas conocidos de AWS Glue

Tenga presente los siguientes problemas conocidos de AWS Glue.

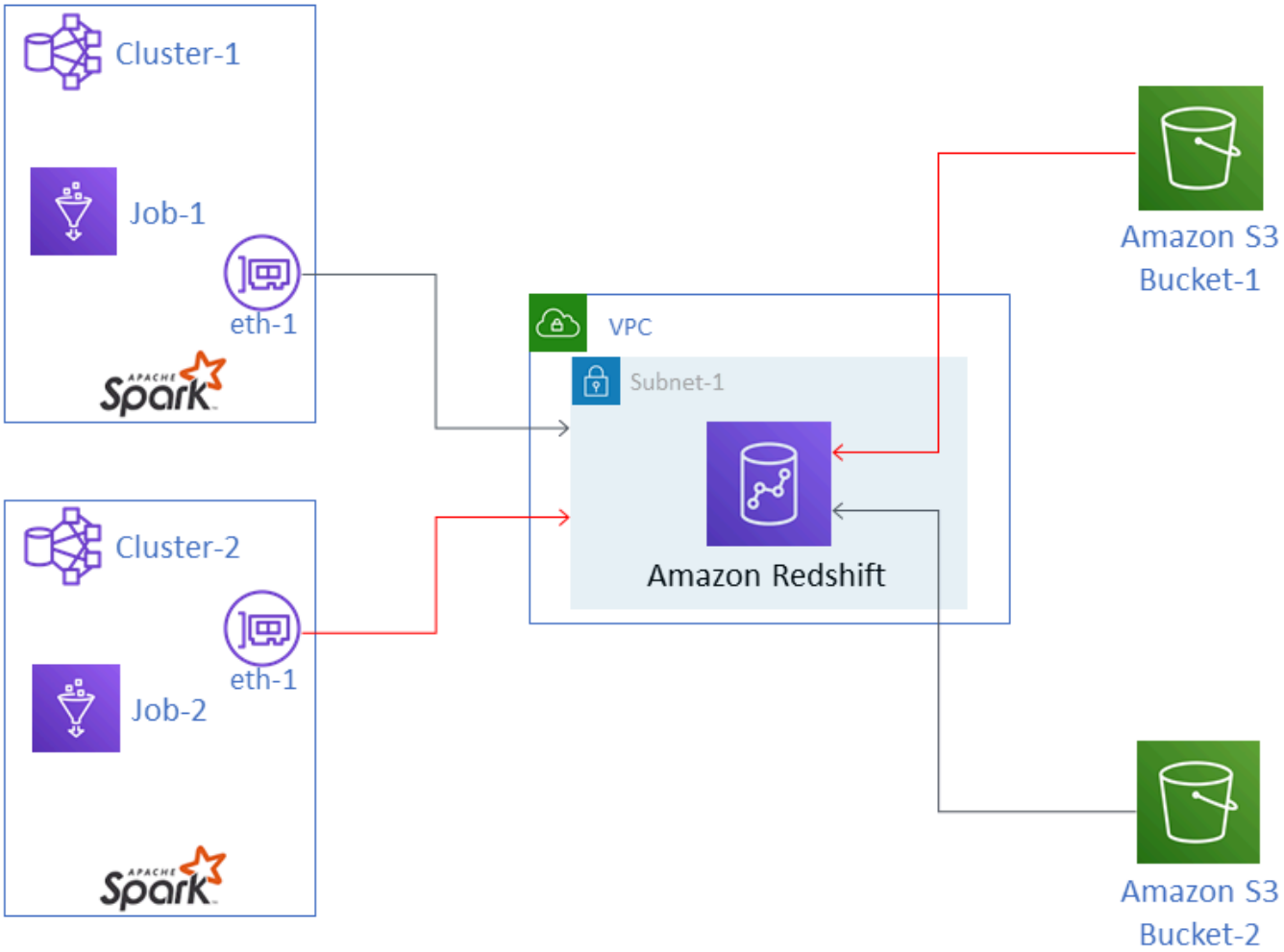
Temas

- Prevención del acceso a datos entre trabajos

Prevención del acceso a datos entre trabajos

Considere la siguiente situación: tiene dos trabajos de AWS Glue Spark en una única cuenta de AWS, cada uno de ellos ejecutándose en un clúster de AWS Glue Spark independiente. Los trabajos utilizan conexiones de AWS Glue para obtener acceso a los recursos de la misma nube virtual privada (VPC). En esta situación, un trabajo que se ejecuta en un clúster podría tener acceso a los datos del trabajo que se ejecuta en el otro clúster.

El siguiente diagrama ilustra un ejemplo de esta situación.



En el diagrama, `Job-1` de AWS Glue se ejecuta en `Cluster-1`, y Job-2 se ejecuta en `Cluster-2`. Ambos trabajos funcionan con la misma instancia de Amazon Redshift, que reside en la `Subnet-1` de una VPC. La `Subnet-1` podría ser una subred pública o privada.

`Job-1` transforma los datos del `Bucket-1` de Amazon Simple Storage Service (Amazon S3) y escribe los datos a Amazon Redshift. `Job-2` hace lo mismo con los datos en el `Bucket-2`. `Job-1` utiliza el rol (IAM) `Role-1` de AWS Identity and Access Management (no se muestra), que otorga acceso al `Bucket-1`. `Job-2` utiliza el `Role-2` (no se muestra), que otorga acceso al `Bucket-2`.

Estos trabajos tienen rutas de red que les permiten comunicarse con los clústeres de los demás y, por lo tanto, obtener acceso a los datos de los demás. Por ejemplo, `Job-2` podría obtener acceso a los datos de `Bucket-1`. En el diagrama, se muestra como la ruta en rojo.

Para evitar esta situación, le recomendamos asociar diferentes configuraciones de seguridad a `Job-1` y `Job-2`. Al asociar las configuraciones de seguridad, el acceso entre trabajos a los datos se bloquea en virtud de los certificados que AWS Glue crea. Las configuraciones de seguridad pueden ser configuraciones ficticias. Es decir, puede crear las configuraciones de seguridad sin habilitar el cifrado de datos de Amazon S3, datos de Amazon CloudWatch o marcadores de trabajos. Las tres opciones de cifrado se pueden deshabilitar.

Para obtener información acerca de las configuraciones de seguridad, consulte the section called "Cifrado de datos escritos por AWS Glue".

Para asociar una configuración de seguridad a un trabajo

1. Abra la consola de AWS Glue en https://console.aws.amazon.com/glue/.
2. En la página Configure the job properties (Configurar propiedades del trabajo) del trabajo, expanda la sección Security configuration, script libraries and job parameters (Seguridad, configuración, bibliotecas de scripts y parámetros de trabajo).
3. Seleccione una configuración de seguridad en la lista.

Historial de documentación para AWS Glue

Cambio	Descripción	Fecha
Support para perfiles AWS Glue de uso	Los administradores pueden crear perfiles de AWS Glue uso para varias clases de usuarios de la cuenta, como desarrolladores, evaluador es y equipos de productos . Esta flexibilidad permite a los administradores aplicar diferentes controles de uso y costes para cada clase de usuarios. Para obtener más información, consulte Configuración de perfiles AWS Glue de uso.	18 de junio de 2024
Support para un conector de Salesforce AWS Glue para Spark	Se agregó información sobre un nuevo AWS Glue conector para Salesforce. Esta función te permite usar Spark AWS Glue para leer y escribir en Salesforce en la AWS Glue versión 4.0 y versiones posteriores. Para obtener más información, consulta Cómo conectarse a Salesforce.	22 de mayo de 2024
Integración de datos de Amazon Q en AWS Glue (GA)	La integración de datos de Amazon Q AWS Glue es una nueva capacidad de IA generativa AWS Glue que permite a los ingenieros de datos y a los desarrolladores	30 de abril de 2024

	de ETL crear trabajos de integración de datos utilizando un lenguaje natural. Los ingenieros y desarrolladores pueden pedirle a Q que cree trabajos, solucione problemas y responda preguntas sobre AWS Glue la integración de datos. Para obtener más información, consulte Integración de datos de Amazon Q en AWS Glue. Esta función incluye una actualización de las `AwsGlueSessionUserRestrictedPolicy` políticas `AwsGlueSessionUserRestrictedServiceRole` AWS gestionadas y de las políticas gestionadas. `AwsGlueSessionUserRestrictedNotebookServiceRole` Para obtener más información, consulte AWS Glue las actualizaciones de las políticas AWS administradas.

Integración de datos de Amazon Q en AWS Glue (versión preliminar)	La integración de datos de Amazon Q AWS Glue es una nueva capacidad de IA generativa AWS Glue que permite a los ingenieros de datos y a los desarrolladores de ETL crear trabajos de integración de datos utilizando un lenguaje natural. Los ingenieros y desarrolladores pueden pedirle a Q que cree trabajos, solucione problemas y responda preguntas sobre AWS Glue la integración de datos. Para obtener más información, consulte Integración de datos de Amazon Q en AWS Glue. Esta función incluye una actualización de la política `AwsGlueSessionUserRestrictedNotebookPolicy` AWS gestionada. Para obtener más información, consulte AWS Glue las actualizaciones de las políticas AWS administradas.	30 de enero de 2024

Actualización de la documentación de AWS Glue Streaming	Se agregó un nuevo capítulo con contenido nuevo y reorganizado para AWS Glue Streaming. Este contenido describe cómo funciona la transmisión AWS Glue, las características del procesamiento de datos en tiempo real y cómo monitorear sus trabajos de transmisión. Para obtener más información, consulte Transmisión de AWS Glue.	27 de diciembre de 2023
Soporte para el uso de una detección de datos confidenciales detallada	La transformación Detectar datos confidenciales ofrece la capacidad de detectar, enmascarar o eliminar entidades definidas por el usuario o predefinidas por AWS Glue. Las acciones detalladas permiten además aplicar una acción específica por entidad. Para obtener más información, consulte Uso de una detección de datos confidenciales detallada.	26 de noviembre de 2023

Support para monitorizar trabajos con métricas de AWS Glue observabilidad	Use las métricas de observabilidad de AWS Glue para obtener información sobre lo que ocurre dentro de sus trabajos de AWS Glue for Apache Spark y así mejorar la clasificación y el análisis de los problemas. Para obtener más información, consulte Monitoreo con las métricas de observabilidad de AWS Glue.	26 de noviembre de 2023
Support para la detección de anomalías en AWS Glue Data Quality	La detección de anomalías en la calidad de los datos de AWS Glue aplica algoritmos de machine learning (ML) a las estadísticas de datos a lo largo del tiempo para detectar patrones anormales y problemas ocultos de calidad de los datos que son difíciles de detectar con reglas. Para obtener más información, consulte Detección de anomalías de calidad de datos de AWS Glue.	26 de noviembre de 2023

Actualización del comportamiento de registro predeterminado de la interfaz de usuario de Spark	Los trabajos de Spark que generen registros de la interfaz de usuario de Spark ahora se escribirán con un patrón de nombre de archivo diferente para que sean compatibles con la interfaz de usuario de Spark en la AWS Glue consola. Esto no cambia el comportamiento de los CloudWatch registros. Puede volver al comportamiento anterior si actualiza la configuración de su trabajo. Para obtener más información, consulte Monitorización de trabajos mediante la interfaz de usuario web de Apache Spark.	17 de noviembre de 2023

Support para nuevas fuentes de datos en AWS Glue Spark	Las conexiones a Amazon OpenSearch Service, Azure SQL, Azure Cosmos for NoSQL, SAP HANA, Teradata Vantage y Vertica ahora se admiten de forma nativa desde dentro. AWS Glue Además, las conexiones a estas fuentes de datos, junto con MongoDB, ahora están disponibles para su uso en AWS Glue el editor visual de Studio. Para obtener más información, consulta Tipos y opciones de conexión para ETL en Spark AWS Glue para obtener información sobre AWS Glue la compatibilidad con Spark y Añadir una AWS Glue conexión para obtener información sobre su uso en el editor visual de AWS Glue Studio.	17 de noviembre de 2023
Soporte para generar estadísticas de columnas	Puedes calcular estadísticas a nivel de columna para AWS Glue Data Catalog tablas en formatos de datos como Parquet, ORC, JSON, ION, CSV y XML sin necesidad de configurar canalizaciones de datos adicionales. Para obtener más información, consulte Trabajar con las estadísticas de las columnas.	16 de noviembre de 2023

Soporte para la compactación de datos para tablas Iceberg	Para mejorar el rendimiento de lectura de los servicios de AWS análisis, como Amazon Athena y Amazon EMR, y los trabajos de AWS Glue ETL, Data Catalog proporciona una compactación gestionada (un proceso que compacta objetos pequeños de Amazon S3 para convertirlos en objetos más grandes) para las tablas Iceberg de Data Catalog. Para obtener más información, consulte Optimización de las tablas de Iceberg.	13 de noviembre de 2023
Actualizar el comportamiento de espera al ejecutar un trabajo	Las ejecuciones de tareas de intérprete de comandos estándar de Spark y Python ahora pasarán a `WAITING` en determinadas situacion es, al contrario de pasar inmediatamente a `FAILED`. Para obtener más información, consulte Estados de ejecución de trabajos de AWS Glue.	8 de noviembre de 2023
Guía del usuario de AWS Glue Studio consolidada en la guía para desarrolladores de AWS Glue	La guía del usuario de AWS Glue Studio se ha trasladado a la guía para desarrolladores para crear una guía de usuario única y unificada para AWS Glue Studio, la consola de AWS Glue y el acceso de AWS Glue Studio mediante programación.	25 de octubre de 2023

Actualización de la política gestionada AWSGlueServiceNotebookRole AWS	Se agregó información sobre una actualización menor de la política AWSGlueServiceNotebookRole AWS administrada. Para obtener más información, consulte AWS GlueActualizaciones de las políticas AWS gestionadas.	9 de octubre de 2023
AWS Glue Studio admite cinco nuevas transformaciones integradas	AWS Glue Studio admite las siguientes cinco transformaciones integradas nuevas: la coincidencia de registros, la eliminación de filas nulas, la columna Parse JSON, la ruta de extracción de JSON y el extractor de expresiones regulares. Para obtener más información, consulte Edición de nodos de transformación de datos AWS Glue gestionados.	11 de agosto de 2023
Actualización de la política AWSGlueServiceRole AWS gestionada	Se agregó información sobre una actualización menor de la política AWSGlueServiceRole AWS administrada. Para obtener más información, consulte AWS GlueActualizaciones de las políticas AWS gestionadas.	4 de agosto de 2023

Compatibilidad para el rastreo de tablas de Apache Hudi	Se agregó información sobre cómo AWS Glue rastrear tablas Hudi en buckets de Amazon S3 y cómo registrar las tablas Hudi en. AWS Glue Data Catalog Para obtener más información, consulte ¿Qué almacenes de datos puedo rastrear? y Propiedades del rastreador.	21 de julio de 2023
Actualización de la política administrada AWSGlueConsoleFullAccess AWS	Se agregó información sobre una actualización menor de la política AWSGlueConsoleFullAccess AWS administrada. Para obtener más información, consulte AWS GlueActualizaciones de las políticas AWS gestionadas.	14 de julio de 2023
Compatibilidad para el rastreo de tablas de Apache Iceberg	Se agregó información sobre cómo AWS Glue rastrear tablas Iceberg en buckets de Amazon S3 y cómo registrar las tablas Iceberg en. AWS Glue Data Catalog Para obtener más información, consulte ¿Qué almacenes de datos puedo rastrear? y Propiedades del rastreador.	7 de julio de 2023

Support for AWS Glue with Ray	Se agregó información sobre AWS Glue Ray, un nuevo motor que puede respaldar AWS Glue trabajos. Se reorganizó el contenido existente AWS Glue con Spark para eliminar la ambigüedad.	30 de mayo de 2023
Support for AWS Glue Data Quality (GA)	AWS Glue La calidad de los datos ahora está disponible de forma general. AWS Glue La calidad de los datos le ayuda a evaluar y supervisar la calidad de los datos. Para obtener información sobre cómo utilizar la calidad de AWS Glue los datos con el catálogo de datos, consulte Calidad AWS Glue de los datos. Para obtener más información sobre la calidad de AWS Glue los datos AWS Glue Studio, consulte Evaluar la calidad de los datos con AWS Glue Studio.	24 de mayo de 2023

Compatibilidad para tipos de trabajos más grandes para trabajos de Apache Spark	Ya se encuentra disponible la asistencia para los tipos de trabajo de `G.4X` y `G.8X` para los trabajos de Apache Spark. Estos tipos de trabajos son adecuados para los trabajos cuyas cargas de trabajo contienen las transformaciones, agregaciones, combinaciones y consultas más exigentes. Para obtener más información, consulte Agregar trabajos en AWS Glue.	8 de mayo de 2023
Compatibilidad para crear índices de particiones cuando se rastrean tablas	Se agregó información sobre cómo los rastreadores admiten la creación de índices de partición para las tablas que detecta el rastreador. Para obtener más información, consulte Establecimiento de opciones de configuración de rastreadores.	24 de abril de 2023
Compatibilidad para métricas de uso de recursos	Se agregó información sobre la visualización del uso de recursos del servicio y la configuración de alarmas en Amazon CloudWatch. Para obtener más información, consulte AWS Glue resource monitoring.	7 de abril de 2023

Actualización de la política AWSGlueConsoleFullAccess AWS gestionada	Se agregó información sobre una actualización menor de la política AWSGlueConsoleFullAccess AWS administrada. Para obtener más información, consulte AWS GlueActualizaciones de las políticas AWS gestionadas.	28 de marzo de 2023
Se agregó una guía para su uso AWS Glue con un AWS SDK con ejemplos	La Guía para AWS Glue desarrolladores incluye dos secciones nuevas que proporcionan información que te ayudará a AWS Glue utilizarla con un AWS SDK. Para obtener más información, consulte Uso AWS Glue con un AWS SDK y ejemplos de código para AWS Glue usar AWS los SDK.	23 de febrero de 2023
Actualice la documentación de IAM con AWS Glue	Se reorganizó y agregó información sobre el uso de IAM con. AWS Glue Para obtener más información, consulte Administración de identidad y acceso para AWS Glue.	15 de febrero de 2023

Soporte para la ejecución de trabajos de ETL de streaming en la versión 4.0 de AWS Glue	Se agregó información sobre la compatibilidad para ejecutar trabajos de ETL de streaming en la versión 4.0 de Glue y sobre las nuevas opciones para conectarse a un clúster de Kafka o a un clúster de Amazon Managed Streaming para Apache Kafka y Amazon Kinesis Data Streams. Para obtener más información, consulte Agregar trabajos de ETL de streaming en AWS Glue y Tipos de conexión y opciones para ETL en AWS Glue.	8 de febrero de 2023
Compatibilidad con el rastreo de orígenes de datos de MongoDB Atlas	Se agregó información sobre cómo rastrear AWS Glue las fuentes de datos de MongoDB Atlas. Para obtener más información, consulte ¿Qué almacenes de datos puedo rastrear?, Propiedades de conexión de MongoDB y MongoDB Atlas y Utilización de la conexión de MongoDB o MongoDB Atlas.	6 de febrero de 2023

Compatibilidad con el rastreo de tablas de Delta Lake mediante un conector nativo de Delta Lake	Se agregó información sobre cómo rastrear AWS Glue las tablas de Delta Lake mediante un conector nativo de Delta Lake. Esta función le permite utilizar los motores de AWS consultas para consultar directamente el registro de transacciones de Delta y utilizar funciones como los viajes en el tiempo y las garantías ACID, y sincroniz ar los metadatos de Delta Lake de los archivos de transacciones de Amazon S3 con el catálogo de datos para habilitar los permisos de columna en sus consultas en Lake Formation. Para obtener más información, consulte Cómo especificar opciones de configuración para un almacén de datos de Delta Lake y Consulta de tablas de Delta Lake.	15 de diciembre de 2022

Support for AWS Glue Data Quality (versión preliminar)	Support ya está disponible para AWS Glue Data Quality (versión preliminar). AWS Glue La calidad de los datos le ayuda a evaluar y supervisar la calidad de los datos cuando utiliza la AWS Glue versión 3.0. Para obtener información sobre cómo utilizar la calidad de AWS Glue los datos con el catálogo de datos, consulte Calidad de AWS Glue datos (vista previa). Para obtener más información sobre la calidad de AWS Glue los datos AWS Glue Studio, consulte Evaluar la calidad de los datos con AWS Glue Studio.	30 de noviembre de 2022
Compatibilidad con un nuevo conector de Spark para Amazon Redshift con nuevas características y mejoras de rendimiento	Ya hay compatibilidad con un nuevo conector de Spark para Amazon Redshift con un nuevo controlador de JDBC que se puede utilizar con tareas de ETL de AWS Glue para crear aplicaciones de Apache Spark que lean datos desde Amazon Redshift y escriban en este como parte de las canalizaciones de ingesta y transformación de datos. Para más informaci ón, consulte Movimiento de datos desde y hacia Amazon Redshift.	29 de noviembre de 2022

Compatibilidad con la versión 4.0 de AWS Glue.	Se agregó información acerca de la compatibilidad con la versión 4.0 de AWS Glue. Las características incluyen la compatibilidad nativa para marcos de lagos de datos abiertos con Apache Hudi, Delta Lake y Apache Iceberg, y la compatibilidad nativa con el complemento Cloud Shuffle Storage basado en Amazon S3 (un complemento de Apache Spark) para utilizar Amazon S3 para una capacidad de almacenamiento aleatoria y elástica. Para obtener más información, consulte las Notas de la versión de AWS Glue y Migración de trabajos de AWS Glue a la versión 4.0 de AWS Glue.	28 de noviembre de 2022
AWS Glue Studio ahora ofrece transformaciones visuales personalizadas	Las transformaciones visuales personalizadas permiten a los clientes definir, reutilizar y compartir la lógica de ETL específica de la empresa entre sus equipos. Para obtener más información, consulte Custom visual transforms (Transformaciones visuales personalizadas).	28 de noviembre de 2022

Compatibilidad con el uso del rastreador de AWS Glue para publicar metadatos de los almacenes de datos de JDBC	Ya está disponible la compatibilidad con el uso del rastreador de AWS Glue para publicar metadatos, como comentarios y tipos sin procesar, en el catálogo de datos de los almacenes de datos de JDBC. Para obtener más información, consulte los parámetros establecidos en las tablas del catálogo de datos por rastreador, las propiedades del rastreador y JdbcTarget la estructura del rastreador.	18 de noviembre de 2022

Compatibilidad con el rastreo de almacenes de datos de Snowflake	Ahora hay compatibilidad con el uso de AWS Glue para rastrear tablas y vistas de Snowflake y publicar los metadatos en Catálogo de datos como una entrada de tabla. En el caso de las tablas externas de Snowflake en Amazon S3, el rastreador también rastrea la ubicación de Amazon S3 y el tipo de formato de archivo de la tabla externa y los rellena como parámetros de la tabla. Para más información, consulte ¿Qué almacenes de datos puedo rastrear?, Propiedades de las conexiones de AWS Glue y Parámetros establecidos en las tablas del Catálogo de datos por el rastreador.	18 de noviembre de 2022
Compatibilidad con la mejora de la gestión aleatoria de las aplicaciones de Spark	Ya está disponible la compatibilidad con un nuevo complemento de Cloud Shuffle Storage para Apache Spark. Para más información, consulte Complemento de mezclas aleatorias de Spark para AWS Glue con Amazon S3 y Complemento Cloud Shuffle Storage para Apache Spark.	15 de noviembre de 2022

Se agregó soporte para los objetivos del Catálogo de datos al acelerar las notificaciones de eventos de Amazon S3	Además del soporte existente para destinos de Amazon S3, ahora se ofrece soporte para acelerar los rastreos de los destinos del Catálogo de datos mediante notificaciones de eventos de Amazon S3. Para obtener más información, consulte Aceleración de los rastreos mediante las notificaciones de eventos de Amazon S3.	13 de octubre de 2022
Posibilidad de especificar el número máximo de tablas que puede crear un rastreador	Ahora es posible especificar el número máximo de tablas que el rastreador tiene permitido crear. Para obtener más información, consulte Cómo especificar el número máximo de tablas que el rastreador tiene permitido crear.	6 de septiembre de 2022
Compatibilidad con Python 3.9 de trabajos de intérprete de comandos de Python en AWS Glue	Ahora hay compatibilidad disponible para ejecutar scripts compatibles con Python 3.9 en trabajos de intérprete de comandos de Python en AWS Glue y por elegir el uso de conjuntos de bibliotecas preempaquetadas. Para obtener más información, consulte Trabajos de trabajos de intérprete de comandos de Python en AWS Glue.	11 de agosto de 2022

Support para ejecutar AWS Glue trabajos no urgentes o urgentes con capacidad sobrante	Ahora hay compatibilidad disponible para la configuración de ejecuciones de trabajos flexibles para trabajos no urgentes, como trabajos de preproducción, pruebas y cargas de datos únicas. Para obtener más información, consulte Agregar trabajos en AWS Glue.	9 de agosto de 2022
Soporte para un nuevo tipo de proceso de trabajo para el streaming de trabajos	Ya se encuentra disponible el soporte para el tipo de proceso de trabajo `G.025X` para trabajos de streaming de bajo volumen. Para obtener más información, consulte Agregar trabajos en AWS Glue.	14 de julio de 2022
Soporte para el uso de Kafka SASL en conexiones AWS Glue	Ya se encuentra disponible el soporte para el uso de Kafka SASL en conexiones de AWS Glue. Para obtener más información, consulte Propiedades de conexión de AWS Glue Kafka para autenticación de clientes.	5 de julio de 2022
Compatibilidad con Apache Kafka Connector para esquemas Protobuf	La compatibilidad con Apache Kafka Connector ya está disponible para esquemas Protobuf. Para obtener más información, consulte Registro de esquemas de AWS Glue.	9 de junio de 2022

Compatibilidad con Auto Scaling para trabajos de AWS Glue (disponible de manera general)	Se ha agregado informaci ón sobre el uso de escalado automático para trabajos en AWS Glue versión 3.0 para escalar dinámicamente los recursos de computación. Para obtener más información, consulte Uso de Auto Scaling para AWS Glue.	14 de abril de 2022
Actualice la documentación de AWS Glue desarrollando y probando scripts de trabajo de AWS Glue	Información reorganizada y agregada sobre los métodos de desarrollo y pruebas disponibles para AWS Glue, incluidas instrucciones para desarrollar con Docker. Para obtener más información, consulte Desarrollo y pruebas de scripts de trabajo de AWS Glue.	14 de marzo de 2022
Agregado de búferes de protocolo (Protobuf) como formato de datos admitido para un AWS Glue Schema Registry	Se agregó información acerca de Protobuf como formato de datos admitido (además de AVRO y JSON). Para obtener más información, consulte Registro de esquemas de AWS Glue.	25 de febrero de 2022
Compatibilidad con tablas de rastreo de Delta Lake	Se agregó información sobre cómo rastrear AWS Glue las mesas de Delta Lake. Para obtener más información, consulte Cómo especificar opciones de configuración para un almacén de datos de Delta Lake.	24 de febrero de 2022

Support for AWS Glue job insights	Se agregó información sobre el uso AWS Glue de la información sobre los trabajos para simplificar la depuración y la optimización de los AWS Glue trabajos. Para obtener más información, consulte Monitorización con información de trabajos de AWS Glue.	8 de febrero de 2022
Compatibilidad para rastrear tablas del Catálogo de datos respaldadas por Amazon S3 mediante un punto de conexión de VPC	Además de los almacenes de datos de Amazon S3, puede configurar las tablas del Catálogo de datos respaldadas por Amazon S3 para acceder a ellas únicamente a través de un entorno de Amazon Virtual Private Cloud (Amazon VPC), con fines de seguridad, auditoría o control. Para obtener más información, consulte Rastreo de un almacén de datos de Amazon S3 o tablas del Catálogo de datos respaldadas por Amazon S3 mediante un punto de conexión de VPC.	3 de febrero de 2022

Compatibilidad con las tablas regidas por Lake Formation	Se ha agregado información acerca de la compatibilidad de AWS Glue con las tablas regidas por Lake Formation , que admiten transacciones ACID, compactación automática de datos y consultas de viaje en el tiempo. Para obtener más información, consulte la API de AWS Glue y la Guía para desarrolladores de AWS Lake Formation.	30 de noviembre de 2021
Se han añadido nuevas políticas AWS gestionadas para las sesiones interactivas y los cuadernos	Las nuevas políticas gestionadas de IAM proporcionaron una mayor seguridad para su uso AWS Glue con sesiones y cuadernos interactivos. Para obtener más información, consulte Políticas administradas de AWS para AWS Glue.	30 de noviembre de 2021
Glue Schema Registry ahora es compatible con trabajos de streaming	Puede crear trabajos de streaming que tengan acceso a las tablas que forman parte de Glue Schema Registry. Para obtener más información, consulte AWS Glue Schema Registry y Agregado de trabajos de ETL de streaming en AWS Glue.	15 de noviembre de 2021

Compatibilidad con las nuevas características de machine learning	Se ha agregado información sobre las nuevas caracterí sticas de la transformación de machine learning de búsqueda de coincidencias, incluidas la coincidencia progresiva y la puntuación de coincidencias. Para obtener más informaci ón, consulte Búsqueda de coincidencias progresivas y Estimación de la calidad de las coincidencias mediante las puntuaciones de confianza de coincidencias.	31 de octubre de 2021
(Versión preliminar privada) Compatibilidad para trabajos flexibles de AWS Glue	Se agregó información sobre la configuración de trabajos Spark de AWS Glue con una clase de ejecución flexible, adecuada para trabajos insensibles al tiempo cuyos tiempos de inicio y finalización pueden variar. Para obtener más información, consulte Agregar trabajos en AWS Glue.	29 de octubre de 2021
Compatibilidad con la aceleración de los rastreos mediante las notificaciones de eventos de Amazon S3	Se ha agregado informaci ón acerca de cómo acelerar los rastreos mediante las notificaciones de eventos de Amazon S3. Para obtener más información, consulte Aceleración de los rastreos mediante las notificaciones de eventos de Amazon S3.	15 de octubre de 2021

Opciones de configuración de seguridad adicionales relacionadas con el control de acceso y las VPC	Se ha agregado información acerca de cómo configurar nuevos permisos de control de acceso en AWS Glue y la configuración de VPC. Para obtener más información, consulte AWSEtiquetasAWS Glue, Políticas basadas en la identidad (políticas de IAM) que controlan la configuración mediante claves de condición o claves de contexto y Configuración de todas las AWS llamadas para que pasen por su VPC.	13 de octubre de 2021
Compatibilidad con las políticas de punto de conexión de VPC	Se ha agregado información acerca de la compatibilidad con las políticas de punto de conexión de Virtual Private Cloud (VPC) en AWS Glue. Para obtener más información, consulte AWS Glue y puntos de conexión de VPC de tipo interfaz (AWS PrivateLink).	11 de octubre de 2021
Glue Studio está disponible en China	Ahora, AWS Glue Studio está disponible en las regiones de China (Pekín) y Ningxia.	11 de octubre de 2021

AWS Glue Studio ofrece creación de cuadernos para editar trabajos interactivos	Los cuadernos le ayudan a escribir y ejecutar código, visualizar los resultados y compartir información. Por lo general, los científicos de datos utilizan cuadernos para experimentos y tareas de exploración de datos. Para obtener más información, consulte Using Notebooks (Uso de cuadernos).	1 de octubre de 2021
Ahora, se encuentra disponible el acceso directo a orígenes de streaming	Al agregar orígenes de datos al trabajo ETL en el editor visual, puede proporcionar información para acceder a la secuencia de datos en lugar de tener que utilizar una base de datos y una tabla del Data Catalog.	30 de septiembre de 2021
Se ha documentado la política de compatibilidad de versiones de AWS Glue	Se ha agregado información acerca de la política de compatibilidad de versiones de AWS Glue y las fases de fin de vida útil para determinadas versiones de AWS Glue. Para obtener más información, consulte Política de compatibilidad de versiones de AWS Glue.	24 de septiembre de 2021

Los conectores personalizados ahora se pueden utilizar con vistas previas de datos	Al editar el nodo de origen de datos mediante un conector personalizado, puede obtener una vista previa del conjunto de datos al elegir la pestaña Vista previa de Dat. Para obtener más información, consulte Conectores personalizados.	24 de septiembre de 2021
Support para sesiones AWS Glue interactivas (vista previa privada)	(Vista previa privada) Se agregó información sobre el uso de sesiones AWS Glue interactivas para ejecutar cargas de trabajo de Spark en la nube desde cualquier Jupyter Notebook. Las sesiones interactivas son el método preferido para desarrollar el código de AWS Glue extracción, transformación y carga (ETL) cuando se utiliza la AWS Glue versión 2.0 o una versión posterior. Para obtener más información, consulte Configuración y ejecución de sesiones AWS Glue interactivas para Jupyter Notebook.	24 de agosto de 2021

Compatibilidad con la creación de flujos de trabajo a partir de esquemas (disponible de manera general)	Se agregó información acerca de la codificación de casos de uso comunes de extracción, transformación y carga (ETL) en proyectos y la creación de flujos de trabajo a partir de proyectos. Permite a los analistas de datos crear y ejecutar con facilidad procesos de ETL complejos. Para obtener más información, consulte Realización de actividades de ETL complejas mediante proyectos y flujos de trabajo en AWS Glue.	23 de agosto de 2021

Compatibilidad con la versión 3.0 de AWS Glue.	Se agregó información acerca del soporte de la versión 3.0 de AWS Glue que admite la actualización del motor Apache Spark 3.0 para ejecutar trabajos de ETL de Apache Spark, y otras optimizaciones y actualizaciones. Para obtener más información, consulte las Notas de la versión de AWS Glue y Migración de trabajos de AWS Glue a la versión 3.0 de AWS Glue. Otras características de esta versión incluyen el administrador de mezclas aleatorias de AWS Glue, un lector CSV vectorizado de SIMD y predicados de particiones de catálogo. Para obtener más información, consulte Administrador de mezclas aleatorias de AWS Glue Spark con Amazon S3, Opciones de formato para las entradas y salidas de ETL en AWS Glue, y Filtrado del lado del servidor mediante predicados de partición de catálogo.	18 de agosto de 2021
AWS GovCloud (US) Region	AWS Glue Studioya está disponible en AWS GovCloud (US) Region	18 de agosto de 2021

Creación de intérprete de comandos de Python disponible en AWS Glue Studio	Al crear un nuevo trabajo, ahora puede elegir crear un trabajo de intérprete de comandos de Python. Para obtener más información, consulte Iniciar el proceso de creación de trabajo y Edición de trabajos de intérprete de comandos de Python en AWS Glue Studio.	13 de agosto de 2021
Support para iniciar un flujo de trabajo con un EventBridge evento de Amazon	Se agregó información acerca de cómo AWS Glue puede ser un consumidor de eventos en una arquitectura basada en eventos. Para obtener más información, consulte Inicio de un AWS Glue flujo de trabajo con un EventBridge evento de Amazon y Visualización de los EventBridge eventos que iniciaron un flujo de trabajo.	14 de julio de 2021
Adición de JSON como formato de datos compatible para AWS Glue Schema Registry	Se agregó información acerca de JSON como formato de datos soportado (además de AVRO). Para obtener más información, consulte AWS Glue Schema Registry.	30 de junio de 2021

Crear trabajos de streaming de AWS Glue sin una tabla del Catálogo de datos	La función de Python `create_data_frame_from_options` o `getSource` para scripts de Scala soporta la creación de trabajos de ETL de streaming que hacen referencia a los flujos de datos directamente, en lugar de requerir una tabla del Catálogo de datos.	15 de junio de 2021
AWS GlueLas transform aciones de aprendizaje automático ahora admiten AWS Key Management Service claves	Puede especificar una configuración o AWS KMS clave de seguridad al configurar las transform aciones de AWS Glue Machine Learning con la consola, la CLI o las AWS Glue API. Para obtener más información, consulte Uso de cifrado de datos con transform aciones de machine learning y API de machine learning de AWS Glue.	15 de junio de 2021
Actualización de la política AWSGlueConsoleFullAccess AWS gestionada	Se agregó información sobre una actualización menor de la política AWSGlueConsoleFull Access AWS administrada. Para obtener más informaci ón, consulte AWS GlueActua lizaciones de las políticas AWS gestionadas.	10 de junio de 2021

Visualice el conjunto de datos del trabajo mientras crea y edita trabajos	Puede utilizar la nueva pestaña Previsualización de datos para un nodo en su diagrama de trabajo para ver una muestra de los datos procesados por ese nodo. Para obtener más informaci ón, consulte Utilizar previsual izaciones de datos en el editor visual de trabajos.	7 de junio de 2021
Compatibilidad con la especific ación de un valor que indique la ubicación de la tabla para la salida del rastreador.	Se agregó información sobre cómo especificar un valor que indica la ubicación de la tabla al configurar la salida del rastreador. Para obtener más información, consulte Cómo especificar la ubicación de la tabla.	4 de junio de 2021
Compatibilidad con el rastreo de una muestra de archivos en un conjunto de datos al rastrear un almacén de datos de Simple Storage Service (Amazon S3)	Se agregó información sobre cómo rastrear una muestra de archivos al rastrear Amazon S3. Para obtener más información, consulte Propiedades del rastreador.	10 de mayo de 2021

Compatibilidad con el escritor de parquet optimizado de AWS Glue	Se agregó información sobre el uso del editor de parqué AWS Glue optimizado DynamicFrames para crear o actualizar tablas con la `parquet` clasificación. Para obtener más información, consulte Creación de tablas, actualización de esquemas y agregado de nuevas particiones en el Catálogo de datos desde trabajos de ETL de AWS Glue y Opciones de formato para entradas y salidas de ETL en AWS Glue.	4 de mayo de 2021
Compatibilidad con contraseñas de autenticación de cliente de Kafka	Se agregó información acerca de cómo los trabajos de ETL de streaming en AWS Glue soportan la autenticación de certificados de cliente SSL con los productores de flujos de Apache Kafka. Ahora puede proporcionar un certificado personalizado al definir una conexión de AWS Glue a un clúster de Apache Kafka, que AWS Glue usará al autenticarse con él. Para obtener más información, consulte Propiedades de conexión de AWS Glue y API de conexión.	28 de abril de 2021

Compatibilidad con el consumo de datos de Amazon Kinesis Data Streams en otra cuenta en trabajos de ETL de streaming	Se agregó información sobre cómo crear un trabajo de ETL de streaming para consumir datos de Amazon Kinesis Data Streams en otra cuenta. Para obtener más información, consulte Agregado de trabajos de ETL de streaming en AWS Glue.	30 de marzo de 2021
Transformación SQL disponible	Puede usar un nodo de transformación SQL para escribir su propia transformación en forma de consulta SQL. Para obtener más información, consulte Uso de una consulta SQL para transformar datos.	23 de marzo de 2021
Compatibilidad con la creación de flujos de trabajo a partir de esquemas (versión preliminar pública)	(Previsualización pública) se agregó información acerca de la codificación de casos de uso comunes de extracción, transformación y carga (ETL) en proyectos y, a continuación, creación de flujos de trabajo a partir de proyectos. Permite a los analistas de datos crear y ejecutar con facilidad procesos de ETL complejos. Para obtener más información, consulte Realización de actividades de ETL complejas mediante proyectos y flujos de trabajo en AWS Glue ().	22 de marzo de 2021

Los conectores se pueden utilizar para destinos de datos	Ahora se admite el uso de un AWS Marketplace conector o un conector personalizado para su destino de datos. Para obtener más información, consulte Creación de trabajos con conectores personalizados.	15 de marzo de 2021
Compatibilidad con las métricas de importancia de columna para transformaciones de machine learning de AWS Glue	Se agregó información sobre la visualización de métricas de importancia de columna cuando se trabaja con transformaciones de machine learning de AWS Glue. Para obtener más información, consulte Trabajar con transformaciones de machine learning en la consola de AWS Glue	5 de febrero de 2021
La programación de trabajos ya se encuentra disponible en AWS Glue Studio	Puede definir programaciones basadas en tiempo para las ejecuciones de trabajo en AWS Glue Studio. Puede utilizar la consola para crear una programación básica o definir una programación más compleja con la sintaxis cron de tipo Unix. Para obtener más información, consulte Programación de ejecuciones de trabajo.	21 de diciembre de 2020

Lanzamiento de conectores personalizados de AWS Glue	Los conectores personalizados de AWS Glue le permiten descubrir y suscribirse a conectores en AWS Marketplace. También introdujimos interfaces de tiempo de ejecución de AWS Glue Spark para conectar conectores creados para Apache Spark Datasource, consulta federada de Athena y API de JDBC. Para obtener más información, consulte Uso de conectores y conexiones con AWS Glue Studio.	21 de diciembre de 2020
Compatibilidad con la ejecución de trabajos de ETL de streaming en AWS Glue versión 2.0	Se agregó información sobre el soporte para la ejecución de trabajos de ETL de streaming en Glue versión 2.0. Para obtener más información, consulte Agregado de trabajos de ETL de streaming en AWS Glue.	18 de diciembre de 2020

Compatibilidad con particiones de cargas de trabajo con ejecución limitada	Se agregó información acerca de habilitar la partición de carga de trabajo para configurar los límites superiores en el tamaño del conjunto de datos o la cantidad de archivos procesados en ejecuciones de trabajos de ETL. Para obtener más información, consulte Partición de carga de trabajo con ejecución limitada.	23 de noviembre de 2020
Compatibilidad con la administración mejorada de particiones	Se agregó información acerca de cómo usar nuevas API para agregar o eliminar un índice de partición a/desde una tabla existente. Para obtener más información, consulte Trabajar con índices de partición.	23 de noviembre de 2020
Compatibilidad con AWS Glue Schema Registry	Se agregó información acerca del uso de AWS Glue Schema Registry para descubrir, controlar y evolucionar los esquemas de forma centralizada. Para obtener más información, consulte Registro de esquemas de AWS Glue.	19 de noviembre de 2020

Compatibilidad con el formato de entrada Grok en trabajos de ETL de streaming	Se agregó información sobre la aplicación de patrones Grok a orígenes de streaming , como archivos de registro. Para obtener más informaci ón, consulte Aplicación de patrones Grok a orígenes de streaming.	17 de noviembre de 2020
Compatibilidad con el agregado de etiquetas a flujos de trabajo en la consola de AWS Glue	Se agregó información acerca de cómo agregar etiquetas al crear un flujo de trabajo mediante la consola de AWS Glue. Para obtener más información, consulte Creación y desarrollo de un flujo de trabajo mediante la consola de AWS Glue.	27 de octubre de 2020
Compatibilidad con ejecucion es de rastreadores progresiv as	Se agregó información sobre el soporte de ejecuciones de rastreadores progresivos, que rastrean sólo las carpetas de Amazon S3 agregadas desde la última ejecución. Para obtener más información, consulte Rastreos progresivos.	21 de octubre de 2020

Compatibilidad con la detección de esquemas para orígenes de datos de ETL de streaming. Compatibilidad con orígenes de datos de ETL de streaming de Avro y Kafka autoadministrado	Los trabajos de extracción, transformación y carga (ETL) de streaming en AWS Glue ahora pueden detectar automáticamente el esquema de los registros entrantes y controlar los cambios de esquema por registro. Ahora se soportan orígenes de datos Kafka autoadministrados. Los trabajos de ETL de streaming ahora admiten el formato Avro en los orígenes de datos. Para obtener más información, consulte ETL de streaming en AWS Glue, Definición de propiedades de trabajo para un trabajo de ETL de streaming y Notas y restricciones para orígenes de streaming de Avro.	7 de octubre de 2020
Compatibilidad con el rastreo de orígenes de datos de MongoDB y DocumentDB	Se agregó información acerca del soporte para rastrear orígenes de datos de MongoDB y Amazon DocumentDB (con compatibilidad con MongoDB). Para obtener más información, consulte Definición de rastreadores.	5 de octubre de 2020

Compatibilidad con la conformidad con FIPS	Se agregó información acerca de los puntos de enlace de FIPS para clientes que necesitan módulos criptográficos validados según FIPS 140-2 al acceder a los datos mediante AWS Glue. Para obtener más información, consulte Conformidad con FIPS.	23 de septiembre de 2020
AWS Glue Studio proporciona una interfaz visual fácil de usar para crear y monitorear trabajos	Ahora puede usar una interfaz sencilla basada en gráficos para componer trabajos que mueven y transforman datos, y ejecutarlos en AWS Glue. Puede utilizar el panel de ejecución de trabajos en AWS Glue Studio para monitorear la ejecución de ETL y asegurarse de que sus trabajos funcionen de la manera pretendida. Si quiere obtener más información, consulte la Guía del usuario de AWS Glue Studio.	23 de septiembre de 2020
Compatibilidad con la creación de índices de tabla para mejorar el rendimiento de las consultas	Se agregó información sobre la creación de índices de tabla para permitir la recuperación de un subconjunto de particiones de una tabla. Para obtener más información, consulte Trabajar con índices de partición.	9 de septiembre de 2020

Compatibilidad con tiempos de inicio reducidos al ejecutar trabajos de ETL de Apache Spark en AWS Glue versión 2.0.	Se agregó información acerca del soporte para AWS Glue, versión 2.0, que proporciona una infraestructura mejorada para ejecutar trabajos de ETL de Apache Spark con tiempos de inicio reducidos, cambios en el registro y soporte para especificar módulos adicionales de Python al nivel del trabajo. Para obtener más información, consulte Notas de la versión de AWS Glue y Ejecución de trabajos de ETL de Spark con tiempos de inicio reducidos.	10 de agosto de 2020
Compatibilidad con la limitación de la cantidad de ejecuciones concurrentes de un flujo de trabajo.	Se agregó información acerca de cómo limitar la cantidad de ejecuciones concurrentes del flujo de trabajo para un flujo de trabajo determinado. Para obtener más información, consulte Creación y desarrollo de un flujo de trabajo mediante la consola de AWS Glue.	10 de agosto de 2020

Compatibilidad con el rastreo de un almacén de datos de Simple Storage Service (Amazon S3) mediante un punto de conexión de VPC	Se agregó información acerca de la configuración de su almacén de datos de Amazon S3 para el acceso únicament e a través de un entorno de Amazon Virtual Private Cloud (Amazon VPC), con fines de seguridad, auditoría o control. Para obtener más informaci ón, consulte Rastreo de un almacén de datos de Amazon S3 mediante un punto de enlace de la VPC.	7 de agosto de 2020
Compatibilidad con la reanudación de ejecuciones de flujos de trabajo	Se agregó información acerca de cómo reanudar las ejecuciones de flujo de trabajo que solo se completar on en forma parcial porque uno o más nodos (trabajos o rastreadores) no se completar on correctamente. Para obtener más informaci ón, consulte Reparación y reanudación de una ejecución de flujo de trabajo.	27 de julio de 2020

Compatibilidad con la habilitación de certificados de CA privados en conexiones de Kafka en AWS Glue.	Se agregó información sobre las nuevas opciones de conexión que soportan la habilitación de certificados de CA privados para conexiones Kafka en AWS Glue. Para obtener más información, consulte Tipos y opciones de conexión para ETL en AWS Glue y Parámetros especiales usados por AWS Glue.	20 de julio de 2020
Compatibilidad con la lectura de datos de DynamoDB en otra cuenta	Se agregó información sobre el soporte de AWS Glue para leer datos de la tabla DynamoDB de otra cuenta de AWS . Para obtener más información, consulte Lectura de datos de DynamoDB en otra cuenta.	17 de julio de 2020
Compatibilidad con la conexión de escritura de DynamoDB en AWS Glue versión 1.0 o posterior	Se agregó información acerca del soporte para el escritor de DynamoDB y opciones de conexión nuevas o actualizadas para que DynamoDB lea o escriba. Para obtener más información, consulte Tipos y opciones de conexión para ETL en AWS Glue.	17 de julio de 2020

Compatibilidad con enlaces de recursos y con el control de acceso entre cuentas mediante AWS Glue y Lake Formation	Se agregó contenido sobre los nuevos objetos del Catálogo de datos denominados enlaces de recursos y sobre cómo administrar los recursos compartidos del Catálogo de datos entre cuentas con AWS Glue y AWS Lake Formation. Para obtener más información, consulte Concesión de acceso entre cuentas y Enlaces de recursos de tabla.	7 de julio de 2020
Compatibilidad con el muestreo de registros al rastrear almacenes de datos de DynamoDB	Se agregó información acerca de las nuevas propiedad es que puede configurar al rastrear un almacén de datos de DynamoDB. Para obtener más información, consulte Propiedades del rastreador.	12 de junio de 2020
Compatibilidad con la detención de la ejecución de un flujo de trabajo.	Se agregó información sobre cómo detener una ejecución de flujo de trabajo para un flujo de trabajo determinado. Para obtener más información, consulte Detener ejecución de flujo de trabajo.	14 de mayo de 2020

Soporte para trabajos de ETL de Spark Streaming	Se agregó información sobre la creación de trabajos de extracción, transformación y carga (ETL) con origen de datos de streaming. Para obtener más información, consulte Agregado de trabajos de ETL de streaming en AWS Glue.	27 de abril de 2020
Compatibilidad con la creación de tablas, la actualización del esquema y la adición de nuevas particiones en el Catálogo de datos después de ejecutar un trabajo de ETL	Se agregó información acerca de cómo permitir la creación de tablas, la actualización del esquema y la incorporación de nuevas particiones para ver los resultados del trabajo de ETL en el Catálogo de datos. Para obtener más información, consulte Creación de tablas, actualización de esquemas y agregado de nuevas particiones en el Catálogo de datos desde trabajos de ETL de AWS Glue.	2 de abril de 2020

Compatibilidad con la especificación de una versión para el formato de datos de Apache Avro como entrada y salida de ETL en AWS Glue	Se agregó información acerca de cómo especificar una versión para el formato de datos de Apache Avro como una entrada y salida de ETL en AWS Glue. La versión predeterminada es 1.7. Puede utilizar la opción de formato `version` para especificar la versión 1.8 de Avro y habilitar la lectura/escritura lógica. Para obtener más información, consulte Opciones de formato para las entradas y salidas de ETL en AWS Glue.	31 de marzo de 2020
Compatibilidad con el confirmador optimizado para S3 de EMRFS para la escritura de datos de Parquet en Simple Storage Service (Amazon S3)	Se agregó información acerca de cómo establecer un nuevo indicador que habilite el confirmador optimizado para S3 de EMRFR de forma que puedan escribirse datos de Parquet en Amazon S3 al crear o actualizar un trabajo de AWS Glue. Para obtener más información, consulte Parámetros especiales utilizados por AWS Glue.	30 de marzo de 2020

Support for machine learning se transforma en un recurso gestionado por etiquetas AWS de recursos	Se ha añadido información sobre el uso de etiquetas de AWS recursos para gestionar y controlar el acceso al aprendizaje automátic oAWS Glue. Puedes asignar etiquetas de AWS recursos a tareas, activadores, puntos finales, rastreadores y transformaciones de aprendiza je automático. AWS Glue Para obtener más información, consulte Etiquetas de AWS en AWS Glue.	2 de marzo de 2020
Compatibilidad con argumento s de trabajo que no se pueden invalidar	Se agregó información acerca del soporte de parámetros especiales de trabajos que no se pueden invalidar en desencadenadores o cuando se ejecuta el trabajo. Para obtener más información, consulte Agregado de trabajos en AWS Glue.	12 de febrero de 2020

Compatibilidad con nuevas transformaciones para trabajar con conjuntos de datos en Simple Storage Service (Amazon S3)	Se agregó información sobre nuevas transformaciones (Merge, Purge y Transitio n) y exclusiones de clases de almacenamiento de Amazon S3 en aplicaciones de Apache Spark para trabajar con conjuntos de datos de Amazon S3. Para obtener más información sobre la compatibi lidad de estas transform aciones en Python, consulte mergeDynamicFrameTrabajar con conjuntos de datos en Amazon S3. Para obtener información sobre Scala, consulte las API mergeDyna micFramesde AWS GlueScala GlueContext .	16 de enero de 2020
Compatibilidad con la actualización del Catálogo de datos con información sobre nuevas particiones desde un trabajo de ETL	Se agregó información sobre cómo codificar un script de extracción, transformación y carga (ETL) para actualiza rlo AWS Glue Data Catalog con la nueva información de particiones. Con esta funcional idad, ya no tendrá que volver a ejecutar el rastreador después de finalizar un trabajo para ver las nuevas particiones. Para obtener más informaci ón, consulte Actualización del Catálogo de datos con nuevas particiones.	15 de enero de 2020

Nuevo tutorial: Uso de un SageMaker cuaderno	Se agregó un tutorial que muestra cómo usar un SageMaker bloc de notas de Amazon para desarrollar sus scripts de ETL y aprendiza je automático. Consulte el tutorial: Utilice Amazon SageMaker Notebook con su terminal de desarrollo.	3 de enero de 2020
Compatibilidad con la lectura desde MongoDB y Amazon DocumentDB (con compatibi lidad con MongoDB)	Se agregó información sobre nuevos tipos de conexión y opciones de conexión para leer y escribir en MongoDB y Amazon DocumentDB (con compatibilidad con MongoDB). Para obtener más información, consulte Tipos y opciones de conexión para ETL en AWS Glue.	17 de diciembre de 2019

Diversas correcciones y aclaraciones	Se han agregado correcciones y aclaraciones en todo el documento. Se han eliminado entradas del capítulo de problemas conocidos. Se han agregado advertencias para indicar que AWS Glue solo soporta claves maestras de cliente (CMK) simétricas al crear configuraciones de seguridad y especificar la configuración de cifrado del Catálogo de datos. Se agregó una nota que indica que AWS Glue no permite escribir en Amazon DynamoDB.	9 de diciembre de 2019
Compatibilidad con controladores JDBC personalizados	Se agregó información sobre la conexión a orígenes de datos y destinos con controladores JDBC que AWS Glue no soporta de forma nativa, como MySQL versión 8 y Oracle Database versión 18. Para obtener más información, consulte Valores ConnectionType de JDBC.	25 de noviembre de 2019

Support para conectar SageMaker ordenadores portátiles a diferentes puntos finales de desarrollo	Se agregó información sobre cómo conectar un SageMaker portátil a diferentes puntos finales de desarrollo. Actualizaciones para describir la nueva acción de la consola al cambiar a un nuevo punto final de desarrollo y la nueva política de SageMaker IAM. Para obtener más información, consulte Trabajar con portátiles en la AWS Glue consola y Crear una política de IAM para Amazon SageMaker Notebooks.	21 de noviembre de 2019
Compatibilidad con la versión de AWS Glue en las transformaciones de machine learning	Se agregó información sobre la definición de la versión de AWS Glue en las transformaciones de machine learning para indicar la versión de AWS Glue con la que son compatibles estas transformaciones. Para obtener más información, consulte Trabajar con transformaciones de machine learning en la consola de AWS Glue.	21 de noviembre de 2019

Compatibilidad con el rebobinado de marcadores de trabajos	Se agregó información sobre el rebobinado de los marcadores de trabajo a cualquier ejecución de trabajo anterior reprocesando los datos solo desde la ejecución del trabajo marcado. Se describen dos nuevas subopciones para la opción `job-bookmark-pause` que le permiten ejecutar un trabajo entre dos marcadores. Para obtener más información, consulte, Seguimiento de los datos procesados mediante marcadores de trabajo y Parámetros especiales usados por AWS Glue.	22 de octubre de 2019
Compatibilidad con certificados JDBC personalizados para conectarse a un almacén de datos	Se agregó información sobre el soporte de AWS Glue con certificados JDBC personalizados para conexiones SSL que tienen orígenes o destinos de datos de AWS Glue. Para obtener más información, consulte Trabajar con conexiones en la consola de AWS Glue.	10 de octubre de 2019

Compatibilidad con archivos wheel de Python	Se agregó información sobre el soporte de AWS Glue con los archivos wheel (junto con los archivos egg) como dependencias para los trabajos de intérprete de comandos de Python. Para obtener más información, consulte Proporcionar su propia biblioteca de Python.	26 de septiembre de 2019
Compatibilidad con el control de versiones de puntos de conexión de desarrollo en AWS Glue	Se ha agregado información sobre la definición de `Glue version` en los puntos de enlace de desarrollo. `Glue version` determina las versiones de Apache Spark y Python compatibles con AWS Glue. Para obtener más información, consulte Añadir un punto de conexión de desarrollo.	19 de septiembre de 2019

Compatibilidad con la supervisión de AWS Glue mediante la interfaz de usuario de Spark	Se ha añadido información sobre el uso de la interfaz de usuario de Apache Spark para monitorizar y depurar trabajos ETL en AWS Glue que se ejecutan en el sistema de trabajos de AWS Glue, así como aplicaciones Spark en puntos de conexión de desarrollo de AWS Glue. Para obtener más información, consulte Monitoreo mediante la interfaz de usuario de AWS Glue Spark.	19 de septiembre de 2019
Se mejoró la compatibilidad con el desarrollo de scripts de ETL locales mediante la biblioteca pública de ETL de AWS Glue	Se ha actualizado el contenido de la biblioteca de ETL de AWS Glue para reflejar que ahora se soporta la versión 1.0 de AWS Glue. Para obtener más información, consulte Desarrollo y prueba de scripts de ETL localmente mediante la biblioteca de ETL de AWS Glue.	18 de septiembre de 2019

Compatibilidad con la exclusión de clases de almacenamiento de Simple Storage Service (Amazon S3) al ejecutar trabajos	Se agregó información sobre la exclusión de clases de almacenamiento de Amazon S3 al ejecutar trabajos de ETL de AWS Glue que leen archivos o particiones desde Amazon S3. Para obtener más información, consulte Exclusión de clases de almacenamiento de Amazon S3.	29 de agosto de 2019
Compatibilidad con el desarrollo de scripts de ETL locales mediante la biblioteca pública de ETL de AWS Glue	Se ha agregado información sobre cómo desarrollar y probar localmente scripts ETL de Python y Scala sin necesidad de una conexión de red. Para obtener más información, consulte Desarrollo y prueba de scripts de ETL localmente mediante la biblioteca de ETL de AWS Glue.	28 de agosto de 2019
Problemas conocidos	Se ha agregado información sobre problemas conocidos en AWS Glue. Para obtener más información, consulte Problemas conocidos de AWS Glue.	28 de agosto de 2019

Compatibilidad con transformaciones de machine learning en AWS Glue	Se ha agregado información sobre las capacidades de machine learning proporcionadas por AWS Glue para crear transformaciones personalizadas. Puede crear estas transformaciones cuando cree un trabajo. Para obtener más información, consulte Transformaciones de machine learning en AWS Glue.	8 de agosto de 2019
Compatibilidad con Amazon Virtual Private Cloud compartida	Se agregó información sobre el soporte de AWS Glue con Amazon Virtual Private Cloud compartida. Para obtener más información, consulte Uso compartido de Amazon VPC.	6 de agosto de 2019
Compatibilidad con el control de versiones en AWS Glue	Se agregó información sobre la definición de `Glue version` en las propiedades del trabajo. La versión de AWS Glue determina las versiones de Apache Spark y Python que soporta AWS Glue. Para obtener más información, consulte Agregar trabajos en AWS Glue.	24 de julio de 2019

Compatibilidad con opciones de configuración adicionales para puntos de conexión de desarrollo	Se ha agregado información sobre las opciones de configuración de los puntos de enlace de desarrollo que tienen cargas de trabajo con uso intensivo de memoria. Puede elegir entre dos nuevas configuraciones que ofrecen más memoria por ejecutor. Para obtener más información, consulte Trabajar con puntos de enlace de desarrollo en la consola de AWS Glue.	24 de julio de 2019
Compatibilidad con la realización de actividades de extracción, transferencia y carga (ETL) mediante flujos de trabajo	Se agregó información sobre el uso de un nuevo elemento denominado flujo de trabajo para diseñar una actividad compleja de extracción, transformación y carga (ETL) de varios trabajos que AWS Glue puede ejecutar como entidad única y realizar su seguimiento. Para obtener más información, consulte Realización de actividades de ETL complejas mediante flujos de trabajo en AWS Glue.	20 de junio de 2019

Compatibilidad con Python 3.6 de trabajos de intérprete de comandos de Python	Se ha añadido información sobre la compatibilidad con Python 3.6 en los trabajos de intérprete de comandos de Python. Puede especific ar Python 2.7 o Python 3.6 como propiedad de trabajo. Para obtener más información, consulte Agregado de trabajos de intérprete de comandos de Python en AWS Glue.	5 de junio de 2019
Compatibilidad con puntos de conexión de nube virtual privada (VPC)	Se agregó información acerca de cómo conectarse directame nte a AWS Glue a través de un punto de enlace de interfaz de la VPC. Cuando se utiliza un punto de enlace de interfaz de la VPC, la comunicación entre la VPC y AWS Glue se realiza en su totalidad y de manera segura dentro de la red de AWS . Para obtener más información, consulte Uso de AWS Glue con puntos de enlace de la VPC.	4 de junio de 2019

Compatibilidad con el registro continuo en tiempo real de los trabajos de AWS Glue.	Se agregó información sobre cómo habilitar y ver los registros de tareas de Apache Spark en tiempo real, CloudWatch incluidos los registros de los controladores, cada uno de los registros de los ejecutores y una barra de progreso de las tareas de Spark. Para obtener más información, consulte la sección Registro continuo de trabajos de AWS Glue.	28 de mayo de 2019
Compatibilidad con tablas existentes del Catálogo de datos como orígenes de rastreador	Se agregó información acerca de la especificación de una lista de tablas existentes del Catálogo de datos como orígenes del rastreador. Los rastreadores pueden detectar los cambios en los esquemas de tabla, actualizar las definiciones de la tabla y habilitar la opción de registrar nuevas particiones como datos nuevos. Para obtener más información, consulte Propiedades del rastreador.	10 de mayo de 2019

Compatibilidad con opciones de configuración adicionales para trabajos con uso intensivo de memoria	Se ha añadido información sobre las opciones de configuración para los trabajos de Apache Spark con cargas de trabajo con uso intensivo de memoria. Puede elegir entre dos nuevas configuraciones que ofrecen más memoria por ejecutor. Para obtener más información, consulte Agregar trabajos en AWS Glue.	5 de abril de 2019
Compatibilidad con clasificadores personalizados de CSV	Se ha añadido información sobre el uso de un clasificador personalizado de CSV para inferir el esquema de distintos tipos de datos CSV. Para obtener más información, consulte Escritura de clasificadores personalizados.	26 de marzo de 2019
Support para etiquetas AWS de recursos	Se agregó información sobre el uso de etiquetas de AWS recursos para ayudarte a administrar y controlar el acceso a tus AWS Glue recursos. Puede asignar etiquetas AWS de recursos a tareas, activadores, puntos finales y rastreadores. AWS Glue Para obtener más información, consulte Etiquetas de AWS en AWS Glue.	20 de marzo de 2019

Compatibilidad con el Catálogo de datos para trabajos de Spark SQL	Se agregó información sobre la configuración de sus AWS Glue tareas y puntos finales de desarrollo para utilizarlos AWS Glue Data Catalog como un Apache Hive Metastore externo. De este modo, los trabajos y los puntos de enlace de desarrollo pueden ejecutar directamente consultas de Apache Spark SQL en las tablas almacenadas en AWS Glue Data Catalog. Para obtener más información, consulte Soporte de AWS Glue Data Catalog para trabajos de Spark SQL.	14 de marzo de 2019
Compatibilidad con trabajos de intérprete de comandos de Python	Se ha añadido información sobre los trabajos de intérprete de comandos de Python y el nuevo campo Capacidad máxima. Para obtener más información, consulte Agregado de trabajos de intérprete de comandos de Python en AWS Glue.	18 de enero de 2019

Compatibilidad con notificaciones cuando se producen cambios en bases de datos y tablas	Se ha añadido información sobre los eventos que se generan para los cambios en las llamadas a la API de base de datos, tabla y partición. Puede configurar acciones en CloudWatch los eventos para responder a estos eventos. Para obtener más información, consulte Automatizar AWS Glue con CloudWatch eventos.	16 de enero de 2019
Compatibilidad con el cifrado de contraseñas de conexión	Se ha añadido información sobre el cifrado de contraseñas usadas en los objetos de conexión. Para obtener más información, consulte Cifrado de contraseñas de conexión.	11 de diciembre de 2018
Compatibilidad con permisos de nivel de recursos y políticas basadas en recursos	Se ha agregado información sobre el uso de permisos de nivel de recursos y políticas basadas en recursos con AWS Glue. Para obtener más información, consulte los temas de Seguridad en AWS Glue.	15 de octubre de 2018
Support para SageMaker ordenadores portátiles	Se agregó información sobre el uso de SageMaker cuadernos con terminales de AWS Glue desarrollo. Para obtener más información, consulte Administración de cuadernos.	5 de octubre de 2018

Compatibilidad con cifrado	Información agregada acerca del uso del cifrado con AWS Glue. Para obtener más información, consulte Cifrado en reposo, Cifrado en tránsito y Configuración del cifrado en AWS Glue.	24 de agosto de 2018
Compatibilidad con métricas de trabajos de Apache Spark	Información agregada acerca del uso de las métricas de Apache Spark para lograr una mejor depuración de los trabajos de ETL y una mejor generación de perfiles en estos. Puede hacer fácilment e un seguimiento de métricas de tiempo de ejecución como, por ejemplo, los bytes leídos y escritos, el uso de la memoria y la carga de la CPU del controlador y los ejecutore s, mientras que los datos se mezclan en forma aleatoria entre los ejecutores desde la consola de AWS Glue. Para obtener más información, consulte Supervisión AWS Glue mediante CloudWatc h métricas, Supervisión y depuración de tareas y Trabajo con tareas en la AWS Glue consola.	13 de julio de 2018

Compatibilidad con DynamoDB como origen de datos	Se agregó información acerca de cómo rastrear y usar DynamoDB como origen de datos de los trabajos de ETL. Para obtener más información, consulte el artículo acerca de cómo catalogar tablas con un rastreador y Parámetros de conexión.	10 de julio de 2018
Actualizaciones para crear un procedimiento de servidor de cuadernos	Se ha incluido información actualizada acerca de cómo crear un servidor de cuadernos en una instancia de Amazon EC2 asociada a un punto de enlace de desarrollo. Para obtener más información, consulte Creación de un servidor de cuadernos asociado a un punto de enlace de desarrollo.	9 de julio de 2018
Actualizaciones ahora disponibles sobre RSS	Ahora puede suscribirse a una fuente RSS para recibir notificaciones sobre actualizaciones de la Guía para desarrolladores de AWS Glue.	25 de junio de 2018
Compatibilidad con notificaciones de retraso de trabajos	Se ha agregado información sobre la configuración de un umbral de retraso cuando se ejecuta un flujo de trabajo. Para obtener más información, consulte Agregar trabajos en AWS Glue.	25 de mayo de 2018

Configurar un rastreador para anexar nuevas columnas	Se agregó información sobre la nueva opción de configuración para los rastreadores,. MergeNewColumns Para obtener más información, consulte Configuración de un rastreador.	7 de mayo de 2018
Compatibilidad con el tiempo de espera de los trabajos	Información agregada sobre la configuración de un umbral de tiempo de espera cuando se ejecuta un flujo de trabajo. Para obtener más información, consulte Agregar trabajos en AWS Glue.	10 de abril de 2018
Compatibilidad con trabajos de desencadenador y script de ETL de Scala basados en estados de ejecución adicionales	Se ha agregado información sobre el uso de Scala como lenguaje de programación de ETL. Además, la API de disparador admite ahora la activación al cumplirse algunas de las condiciones (además de todas las condiciones). Además, los trabajos se pueden activar según una ejecución de flujo de trabajo "con error" o "detenida" (además de una ejecución de flujo de trabajo "de éxito").	12 de enero de 2018

Actualizaciones anteriores

En la siguiente tabla, se describen los cambios importantes que se han realizado en cada versión de la Guía para desarrolladores de AWS Glue anteriores a enero de 2018.

Cambio	Descripción	Fecha
Admita orígenes de datos XML y una nueva opción de configuración del rastreador	Se ha agregado información acerca de la clasificación de los orígenes de datos XML y la nueva opción del rastreador para los cambios en la partición.	16 de noviembre de 2017
Nuevas transform aciones, soporte para motores de bases de datos de Amazon RDS adicionales y mejoras de punto de enlace de desarrollo	Se agregó información acerca de las transformaciones de mapeo y filtrado, el soporte para Microsoft SQL Server de Amazon RDS y Oracle de Amazon RDS, y nuevas características de los puntos de enlace de desarrollo.	29 de septiembre de 2017
Versión inicial de AWS Glue	Esta es la versión inicial de la Guía para desarrolladores de AWS Glue .	14 de agosto de 2017

AWS Glosario

Para obtener la AWS terminología más reciente, consulte el AWS glosario de la Glosario de AWS Referencia.

Las traducciones son generadas a través de traducción automática. En caso de conflicto entre la traducción y la version original de inglés, prevalecerá la version en inglés.